

全国计算机技术与软件专业技术资格（水平）考试参考用书

系统分析师考前辅导
系统分析与设计

全国计算机专业技术资格考试办公室推荐

张友生 王勇 主编　　希赛IT教育研发中心 组编

根据2009版大纲编写



清华大学出版社

全国计算机技术与软件专业技术资格（水平）考试参考用书

系统分析师考前辅导——系统分析与设计

张友生　王勇　主编

希赛 IT 教育研发中心　组编

清华大学出版社

北京

内 容 简 介

本书在参考和分析历年考试试题的基础上，着重对新版的考试大纲规定的内容重点地细化和深化，内容涵盖了系统分析师考试大纲和培训指南（2009 版）中信息系统分析设计案例部分的所有知识点，包括系统计划、需求分析与定义、系统设计、软件设计、软件测试、系统运行与维护；系统可靠性分析与设计、系统安全性和保密性设计、嵌入式系统设计；文档编制、项目管理、企业信息化战略与实施。读者通过本书可以掌握当前主流的系统分析与设计技术，掌握各种系统的设计思想和方法。

本书由希赛 IT 教育研发中心组织编写，作为计算机技术与软件专业技术资格（水平）考试参考用书，同时也可作为系统分析师和系统架构设计师日常工作的参考手册，作为软件设计师和数据库系统工程师进一步深造和发展的必读书籍，也是计算机专业教师的教学和工作参考书。

图书在版编目(CIP)数据

系统分析师考前辅导——系统分析与设计 / 张友生，王勇主编. —北京：清华大学出版社，2009.8
（全国计算机技术与软件专业技术资格（水平）考试参考用书）
ISBN 978-7-302-20594-4

Ⅰ. 系…　Ⅱ. ①张…②王…　Ⅲ. 软件工程-系统分析-工程技术人员-资格考核-自学参考资料　Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字（2009）第 117044 号

责任编辑： 柴文强　赵晓宁
责任校对： 徐俊伟
责任印制：
出版发行： 清华大学出版社　　**地　址：** 北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　编：** 100084
社 总 机： 010-62770175　　**邮　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　销： 全国新华书店
开　本： 185×230　**印　张：** 21　**防伪页：** 1　**字　数：** 484 千字
版　次： 2009 年 8 月第 1 版　**印　次：** 2009 年 8 月第 1 次印刷
印　数：
定　价： 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：010-62770177 转 3103　产品编号：033359-01

前　言

系统分析是IT组织开发优秀的应用系统的重要工作，需要拥有扎实的理论知识和丰富的实际经验的人员来完成。随着应用系统规模越来越大，复杂程度越来越高，系统分析师在系统开发的过程中，发挥着越来越重要的作用。

1．目的

系统分析与设计技术是系统分析师的“看家本领”，也是系统分析师考试的重点和难点之所在。鉴于此，希赛IT教育研发中心组织CSAI顾问团有关专家，在清华大学出版社的大力支持下，编写和出版了本书，作为系统分析师考试的参考用书。期望通过本书，不仅能帮助考生顺利通过考试，更重要的是帮助考生掌握当前的系统分析与设计技术，掌握各种系统的设计思想和方法，把这些技术和方法应用到自己的工作实践中。

2．内容

本书对当前比较主流的系统分析与设计技术进行了讨论。

第1章主要讨论系统计划的提出与选择，可行性研究与效益分析，定义问题与归结模型，新旧系统的分析和比较，系统方案的制定、评价和改进。

第2章主要讨论软件需求与需求工程，包括需求的分类、需求获取的方法、需求分析的任务，以及流行的需求分析方法论。

第3章主要讨论处理流程设计（工作流设计），系统文件设计，数据库的选择与设计，分布式系统的设计，系统运行环境的集成与设计。

第4章主要讨论软件设计的基本原则，结构化设计，面向对象设计，用户界面设计和设计评审。

第5章主要讨论软件测试用例的设计，软件测试的策略和步骤，以及自动化软件测试和面向对象的软件测试。

第6章主要讨论软件维护的实施和管理、系统的扩展和集成、新旧系统的转换和交接，以及系统日常运行管理和服务质量评价。

第7章主要讨论系统的可靠性分析与设计，系统的故障模型和可靠性模型，系统的可靠性分析和可靠度计算，以及提高系统可靠性的措施。

第8章主要讨论系统的安全性和保密性设计，访问控制技术，数据机密性，数据完整性，通信与网络的安全性，以及系统安全管理与安全工程。

第9章主要讨论了嵌入式系统分析与设计，包括嵌入式系统开发的特点和要求、嵌入式系统的基本架构、嵌入式操作系统，以及嵌入式系统开发的相关问题。

第10章主要讨论软件文档的编制及作用。

第 11 章简单而系统地介绍了项目及项目管理的基本概念和方法，尤其强调了软件的质量管理。

第 12 章介绍了企业信息化规划，信息系统建设、信息资源管理，企业信息化的实施，以及管理咨询、知识管理和 CIO 相关知识。

与本书的 2005 版相比，这次改版升级工作主要删除了一些相对陈旧的知识点，根据最新的考试大纲和培训指南（2009 版）进行了内容调整。同时，对已有的内容根据读者的反馈意见进行了部分修订。

3. 作者

本书由希赛 IT 教育研发中心组编，由张友生和王勇主编，参加写作的人员均来自 CSAI 顾问团和希赛 IT 教育研发中心。

全书共分 12 章，第 1 章由张友生、吴小军编写，第 2 章由张友生、徐锋编写，第 3 章由桂阳、简亮编写，第 4 章由王勇、张友生编写，第 5、7 章由陈建忠编写。第 6 章由相红利编写。第 8 章由周峻松编写，第 9 章由邓子云、彭雪阳编写，第 10 章由施游、徐锋编写，第 11 章由田俊国编写，第 12 章由刘兴编写。

4. 致谢

在本书出版之际，要特别感谢 CSAI 顾问团的专家们，因为有了他们的无私奉献和积极参与，才使本书有可能面世，并且逐步完善。同时，本书在编写的过程中，参考了许多高水平的资料和书籍（详见各章的参考文献列表），在此，作者对这些参考文献的作者表示真诚的感谢。

感谢清华大学出版社柴文强老师，他在本书的策划、选题的申报、写作大纲的确定，以及编辑、出版等方面，付出了辛勤的劳动和智慧，给予了作者很多的支持和帮助。

感谢希赛教育的系统分析师学员，正是他们的想法汇成了本书的源动力，他们的意见使本书更加贴近读者。

5. 交流

由于作者水平有限，且本书涉及的知识点较多，书中难免有不妥和错误之处。作者诚恳地期望各位专家和读者不吝指教和帮助，对此，作者将深为感激。

有关本书的反馈意见，读者可在希赛教育网（http://www. educity.cn）论坛“书评在线”版块中的“希赛 IT 教育研发中心”栏目与作者交流，希赛教育的专家们会及时地在线解答读者的疑问。

希赛 IT 教育研发中心

2009 年 4 月

目　　录

第 1 章　系 统 计 划

“预则立，不预则废”，任何成功的始点就是计划。在信息系统建设中，系统计划主要描述从项目提出、选择到确立的过程，包括系统项目的提出与可行性分析，系统方案的制订、评价和改进，遗留系统的评价和处理策略，以及现有软件、硬件和数据资源的有效利用等问题。

1.1　项目的提出与选择

企事业单位和政府机构（以下统称为“企业”）在信息化的过程中，可能基于各种动机提出信息系统项目（包括软件项目、网络项目和系统集成项目等各类信息化项目，以下统称为“项目”或“软件项目”）的建设，有关人员要根据这些动机，确定系统的工作范围，提出系统选择方案，给出选择结果。

1.1.1　项目的立项目标和动机

企业在其自身的运营、管理过程中，对于信息系统项目的建设可能具有多种动机，通常可归结 4 种模式，分别是进行基础研究、进行应用研发、提供技术服务、产品的使用者。

1．进行基础研究

此类项目通常由大学、科研院所、企业集团从事基础研究的部门提出和实施。小规模的研究团队可能仅仅是企业中的一个从事研发工作的部门，中等规模的研究团队可以是研究所或研究院等类似的独立建制的单位，大规模的研究团队可以是国家“863”计划这样跨行业、跨地域协作的国家级研究项目组织。

此类项目的目标通常不仅仅包含对某种产品实现机制、核心技术支撑理论或理论体系的深入钻研，而且也代表着对前沿技术的追踪和对技术发展趋势的早期研判。因此，通常也称为“基础研究”。此类研究通常都被看作一种长期的战略性投资，目标不是为了短期的市场收益和支持当前的市场或行业应用，而是为了开拓未来的市场，创造全新概念的产品、产业或生活方式，建立企业、行业甚至国家的竞争优势。

基础研究更多体现为一种探索性研究，成果多体现为某种理论体系和技术成果。基础研究的工作方式通常是：研究者设想未来的技术趋势、社会环境和人的习惯变迁，大胆构思一种超前的需求，并为满足这种需求而预研某种前沿技术。这样的研究通常没有具体的产品发布目标，也没有苛刻的时间限制，甚至连阶段性目标和长期目标也是由研究人员自己来设定的。在研究过程中需要研究人员充分发挥想象力和创造力，突破现有

理论或技术模型的框架，提出全新的理论体系和技术或产品。

2．进行应用研发

此类项目通常由企业进行立项和开发，企业立项的基本动机是得到应用产品，并向目标客户群进行销售，从而占有市场份额并获取利润。产品一般会基于某类特定客户群体的需求进行设计，有明确和具体的研发目标需求，有严格的时间限制、资源预算，大多以项目方式进行组织，可归入“应用研发”型产品。

“应用研发”型的产品具有一定的通用性客户，通常可能是面向个人消费者的工具软件（例如，办公软件、杀毒软件、游戏软件、共享软件或自由软件等均属于这个范畴）、面向特定领域的工具软件（例如，SQL Server 数据库、AutoCAD 工程绘图软件、Rose 建模工具软件等），也可能是面向特定行业中具有一定普遍适用性的业务、可作为产品进行销售的企业级软件系统（例如，企业资源计划系统、客户关系管理系统、新闻发布系统、人力资源管理系统等）。

3．提供技术服务

对此类项目进行立项的企业通常能向目标客户群提供比较全面的技术服务，而不是单一的软件产品。企业的服务范围可能包含：提供技术和解决方案的咨询，利用现有产品进行系统集成和服务，面向特定客户的软件项目定制开发，对现有的软件系统进行升级和改造，提供软件应用相关的技术支持、服务和培训等。一个企业可以提供上述服务中的一个或多个内容。这些企业通常可能以系统集成商、软件项目定制开发商、咨询商、整体解决方案提供商等各种角色出现。

总的来说，此类企业通常会面向一个特定行业，具有相对稳定的客户群体，具有系列化的软件产品和基于这些产品的技术解决方案，企业对自己所处的应用领域有比较深刻的理解，能够整合技术、产品、方案和应用，通过提供一种综合性的技术服务，而不是单一软件产品，来占有市场份额和获取比提供软件产品更高的利润。此类企业可以看作“技术服务”导向的机构。

4．产品的使用者

产品的使用者是最终客户。对他们来说，项目的立项动机既不是得到软件产品进行销售，也不是为了提供技术服务，而是通过采购产品或技术服务来得到使用价值。例如，个人消费者购买绘图软件是为了存储和处理个人数码相机中的照片，而一个企业通过实施企业资源计划（Enterprise Resource Planning，ERP）系统可能是为了达到控制生产能力、科学计划生产、提高管理水平、获取新的决策能力、降低库存成本、提高资金周转率、建立面向市场订单生产方式等目标，并期望通过这些目标的实现来增强企业竞争力，获取更大的市场份额。对信息技术的使用者来说，信息技术是一种手段，同时也是一种成本。如何用最小的成本和风险获得满意的效果是用户最关心的问题。

产品的使用者可能采用各种方式来进行项目立项，可能是直接采购现有市场上的软件进行使用，也可能是寻求内部或外部能够提供技术服务的企业进行定制开发。

1.1.2　项目立项的价值判断

不同的信息系统项目，立项动机和获益目标也是不同的，并不存在一个统一的项目提出模式。但是，能否达成一个成功的立项，总是取决于人们对项目收益预期的价值判断。

1．对技术的态度

不同类型的信息系统项目立项，具有截然不同的价值观和侧重点，如图 1-1 所示。通常以基础研究为目标的项目是高度技术研究导向的，以应用产品开发为目标的项目重点关注的是技术在具体领域中的应用和推广，而以技术服务为目标的项目则是高度客户业务导向或客户满意导向的，产品的最终客户则主要关注软件的使用、影响和代价等应用性问题。

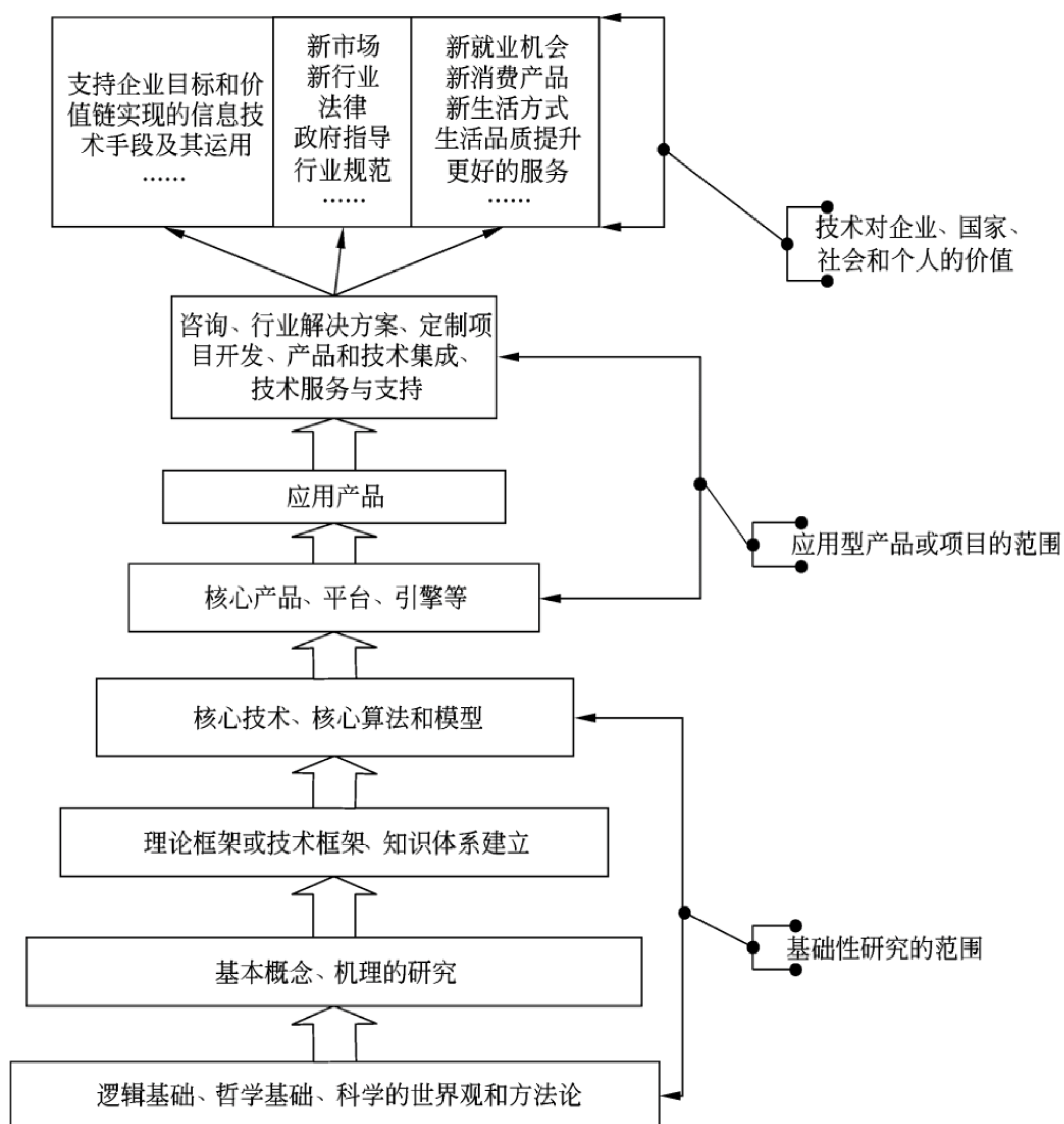


图 1-1　不同软件项目立项对 IT 技术运用的层次

这些价值观彼此之间并不矛盾，只是使用 IT（Information Technology，信息技术）技术的程度不同，对于信息系统项目预期价值的视角不同。但这些对项目基本的价值判断，决定了系统分析师在项目从立项到完成的全过程中，需要长期重点关注的问题侧重点所在，以及需要运用技术手段的程度问题。

2．企业对项目的视角

从企业的角度来看待信息系统项目立项，项目并不是一个简单的、通过技术开发来得到软件产品和完成项目的过程。企业对项目的视角如图 1-2 所示。

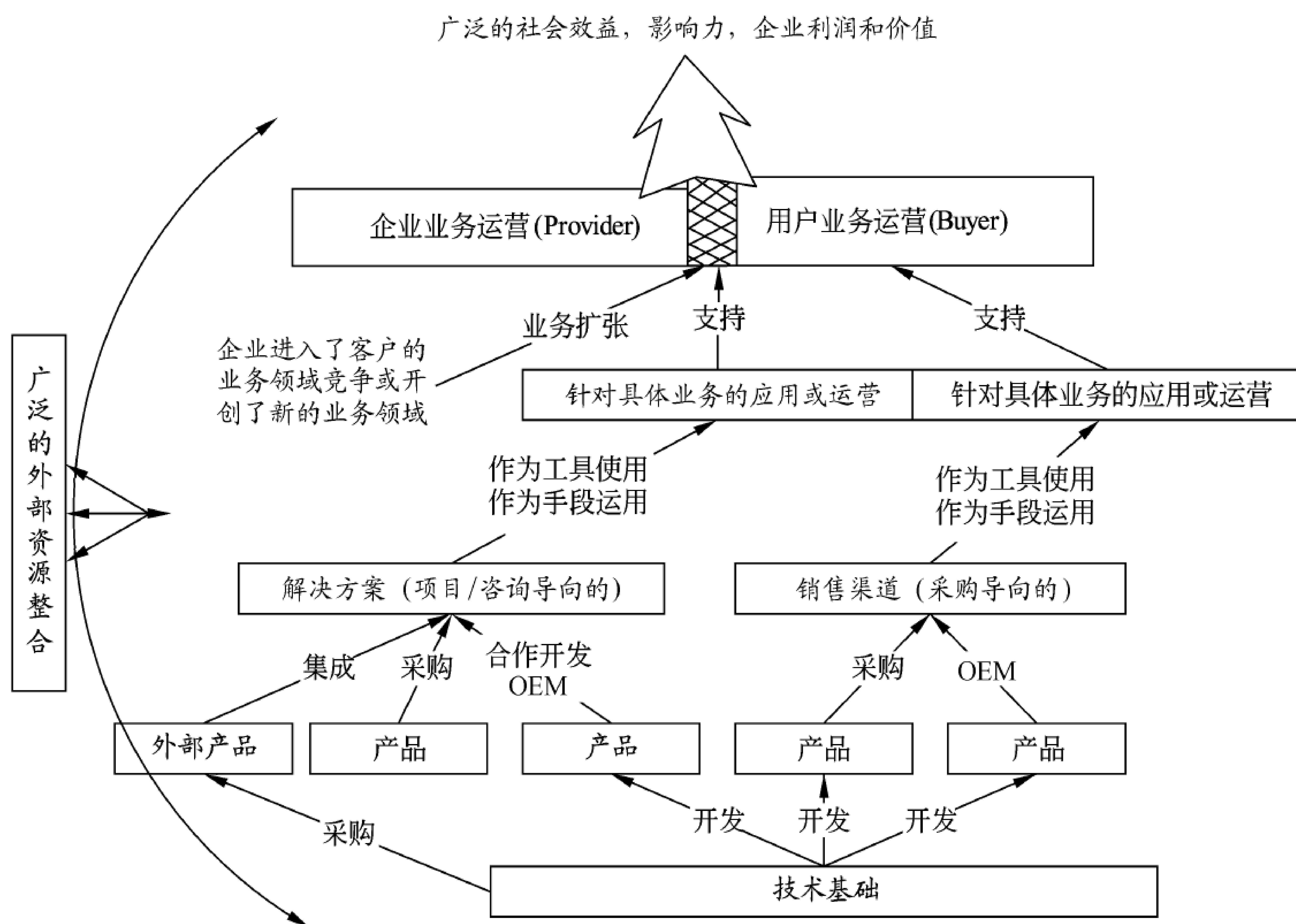


图 1-2　企业对项目或产品的视角

通常，企业总是通过开发产品、提供技术解决方案、整合外部资源、提供咨询和技术服务、销售或运营、进入买方价值链或开创新的领域这 6 个层面来获得价值和利润，图 1-2 中楷体字的部分是对企业最有价值的部分，也是软件所重点针对的领域。

技术、产品、解决方案、技术咨询和服务、资源整合、销售和运营、对业务的理解、业务扩张之间是具有层次关系的，通常，企业得到前者作为基础才能去谋求后者，而企业拥有上述产品层面的要素越多，产品就越有竞争力。根据企业定位不同，或企业所处的时期不同，可能扮演不同的角色。

从企业发展历程来看，企业总是以研发技术、开发软件和销售软件为基础（软件开

发商），然后发展出某方面的技术解决方案（技术解决方案提供商），再然后深入到客户的业务中去解决一些具体的应用（例如，定制项目、系统集成等），最终由于对具体业务的理解和应用能力超越了其所服务的客户，从而能够引领该行业信息化建设（咨询商），或将业务扩张到原用户所经营的范围，成为后来的竞争者（业务扩张），甚至开创出全新的应用领域（运营商）。

企业并不把软件立项和软件产品开发完成看做获取价值的终点，而仅仅是一个起点。不同的自身定位或者不同的时期，企业看待软件产品的价值和作用也截然不同。企业最终的目标可能是获取利润、占有市场份额、加强影响力、广泛的社会效益等这些潜在的商业价值目标，软件则常被作为支持性手段来支撑这些目标的实现。

以提供软件产品为目标的企业，规划的范围一般局限在软件的功能/性能本身，而以运营为目标的企业，规划的范围需涵盖产品、技术方案、业务和运营等各方面。在很多情况下，系统分析师需要超越技术开发的范畴去考察 IT 建设后的目标问题，以便确定软件项目的工作范围、开发边界、项目的阶段性目标，以及未来软件需求变更的根源。

3．系统分析师在项目中的个人角色和工作范围

图 1-3 列举了系统分析师在软件项目中的个人角色和工作范围。

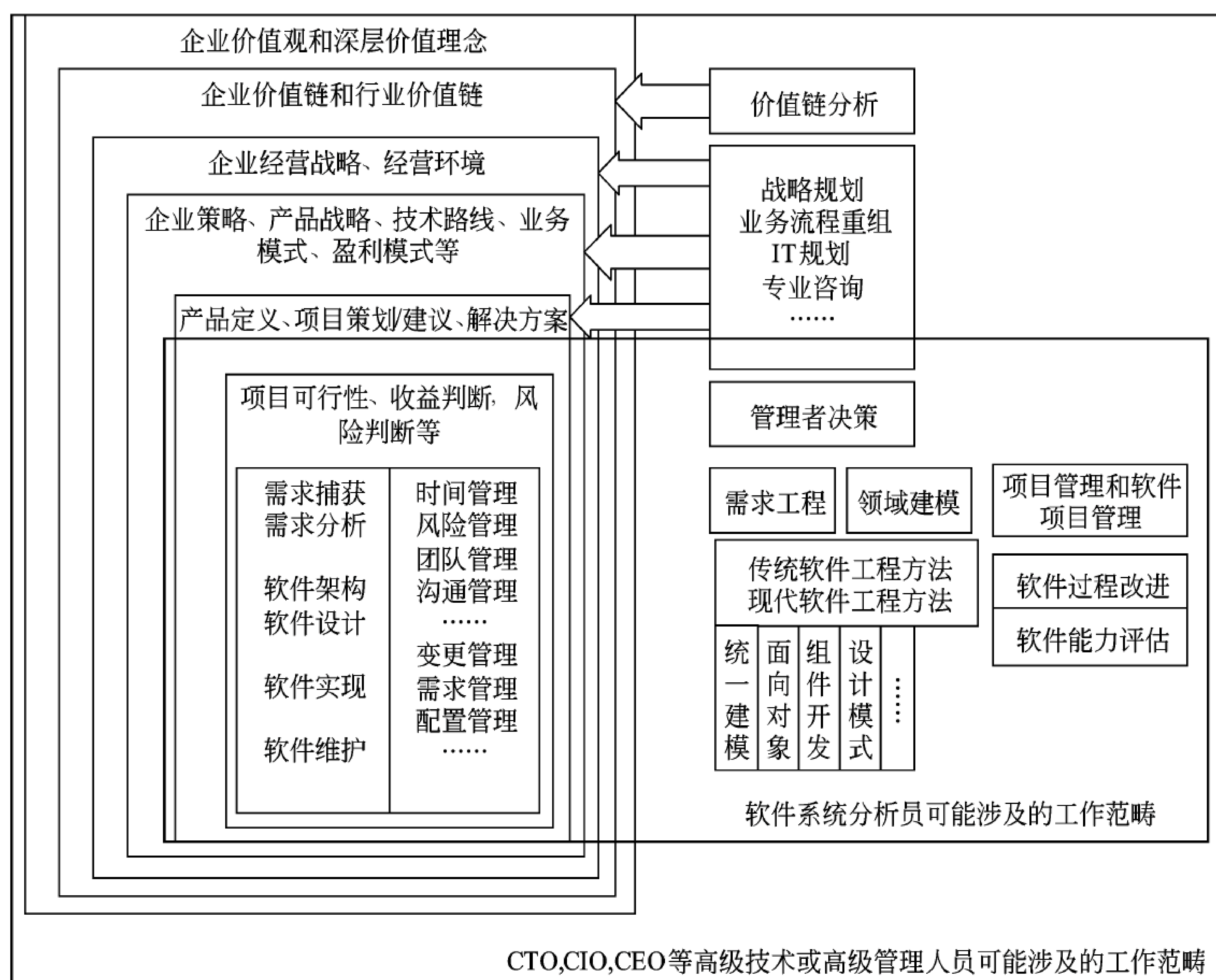


图 1-3　系统分析师的工作范畴

从图 1-3 中可以看到，处于执行层的软件系统分析师，一般仅工作在项目可行性论证、软件架构、软件设计、项目管理和组织等层面上，基本的技术手段是各类软件工程的技术方法和项目管理方法。而处理产品定义、规划、技术路线、业务和赢利模式等的中高层技术人员或管理人员，还必须在更广泛的层面上，对软件项目立项的效益、价值、业务模式、影响、企业战略或策略等进行研究，并将这些作为软件立项的初始需求、发展的需求、未来的需求贯彻到软件立项的目标之中。

希赛教育专家提示：全国计算机技术与软件专业技术资格（水平）考试中的“系统分析师”不仅仅局限于“软件系统分析师”的范畴，而是整个信息系统的系统分析师，即包括软件系统、信息网络系统和系统集成。不过，从考试大纲来看，以软件系统为主。

4．位于战略和执行边界上的系统计划

从软件工程的定义上看，软件工程是一门研究如何使用系统化、规范化、数量化等工程原则和方法去进行软件的开发和维护的学科，其主要研究内容是软件开发技术（包括软件开发方法学、软件工具、软件工程环境）和软件项目管理（包括软件度量、项目估算、进度控制、人员组织、配置管理、项目计划等）。

如果把人们分析问题和解决问题的过程看做 4 个 W 和一个 H，即 What、Why、When、Who、How 的过程，则软件工程解决的主要是执行问题（When、Who 和 How，即什么时候、由谁来做和怎么做的问题），而产品和项目定义阶段主要关注于要解决的目标问题（What 和 Why，即做什么和为什么要这么做）。

系统分析师经常感到困惑的问题是，企业中高层经理指出了一个具体的企业策略（例如，在本年度内要通过信息化手段改善管理水平，降低管理成本），但并没有告知企业将如何去执行这个策略。同时，业务人员、管理人员和从事软件开发的 IT 人员彼此之间对对方的工作、工作的意义所在并不能完全理解。不管怎么样，事情总是需要从沟通开始。因此，系统分析师通常准备了一系列的问题询问业务人员。例如，谁提出了软件项目、最终用户是谁、想象中的操作方式如何等；而业务人员通常不会详细介绍项目背景以及项目背景之后的企业策略问题，往往会直接提出一些很抽象的要求。例如，软件应该界面友好、简洁方便、性能好等。这种通过调研直接进入需求获取或原型化开发的工程手段局限性，可能导致项目需求偏离预期目标，软件需求模糊不清，或未来的软件项目目标不断发生变更。

由于软件工程研究的领域是 How 的问题，而涉及目标的 What 与 Why 的问题无法由软件工程领域的技术手段予以解决。项目提出这个问题域所覆盖的内容，实际上超出了多数软件系统分析师的工作范畴，也部分超出了软件工程本身的范畴，位于战略和执行的边界上。

5．系统计划中的软件规划

为了保证软件项目目标的正确性，在系统计划的阶段，企业经营管理层、系统分析师、客户，以及其他的软件项目建设者需要充分沟通和协调，通过软件项目/产品规划的

手段来保证软件的商业价值目标和软件需求的功能或性能相一致。通过将产品规划的手段与软件工程手段相结合，软件项目开发就能够实现从“面向需求驱动的软件开发过程”向“面向商业价值的软件规划和开发过程”升级。保证企业能够以最小的代价开发最有价值的系统。

因此，担任技术负责人的系统分析师必须积极参与到软件立项背后的产品规划、技术战略、项目远景目标和价值判断的确认过程中。从获取用户需求更进一步去观察项目价值、项目目标，以及项目背后的企业策略问题，辅助企业的经营管理层勾画项目远景目标和项目实施的路线蓝图，完成项目计划到实施的最终决策过程，并最终通过合理确定软件项目的开发边界，规避那些因开发目标错误导致的根源性失败。

在这个过程中，项目需求的初期调研结果，产品或项目未来的商务模式和赢利模式，用户业务描述和想象，用户群体和组织变更，行业或技术的发展趋势，软件针对的用户业务规划，项目资源的投入计划和变更，软件企业自身的产品战略和技术路线，客户和企业所在的行业价值链，来自社会、政策、经济发展等各方面的影响因素均在考察的范围之中。通过这些多角度的分析和考察，总是可以寻找到很多新产品的立项方向和新业务的发展机会，评估已经确立的软件项目是否合理，解释软件项目立项的根本性价值问题。同时，也有助于系统分析师最终定义软件产品或项目的开发边界时，理解和把握软件产品中对最终项目或产品成功具有至关重要影响的功能需求。

1.1.3　项目的选择和确定

信息系统项目的选择和确定，是项目投资方看待项目的视角。具体的项目选择至少包含两种方式。一种是在软件开发企业在诸多的产品方向中选择适当的方向进行研究和开发，另一种是客户从诸多的软件产品和方案中进行选择采购。

与项目提出的问题一样，并不存在一个统一模式进行项目的选择和取舍，但存在一些进行项目取舍和评估的基本原则，通过使用这些原则，可以逐步排除那些不符合需求的项目定义，选择和确定满意的项目或产品。项目选择和确定的原则与系统计划所使用的软件规划原则是一致的，核心问题都是软件项目的目标规划问题。

1．选择有核心价值的产品/项目或开发方向

这个策略的关键在于确定什么样的项目是有价值的。通常，由于立项单位所处的行业、在行业中的位置、立项目标等因素不同，对软件项目的价值判断也不同。但是，一般来说，有核心价值的软件项目通常总是和企业或客户的核心业务相关。

美国哈佛商学院的著名教授 Michael Porter 曾经在他的《竞争优势》（Competitive Advantage）一书中提出了价值链的概念，价值链把企业运作的各种活动划分为产品设计、生产、营销和应用等独立领域，企业的价值链也可以进一步和上游供货商与下游买主的价值链相连，从而构成一个产业的价值链。如果以价值链的观点来看待软件产品或项目，软件是作为一种技术服务手段被作用到企业业务的价值链上的。软件手段通过实现业务

价值链中关键业务的信息化，用改善甚至重建的方式提升这些业务的运行质量或效率。

在企业经营活动中，对价值链增值最大的部分就是企业的核心业务。针对核心业务的信息化产品或项目，通常都是具有高价值的，也可以说，信息化的关键就是这些核心业务的信息化。举例如下：

（1）对生产制造业的企业来说，管理和调度企业资源、生产计划、库存控制、实现面向订单（面向市场）的生产和销售等方面就是核心业务，无论实施 ERP 这种大规模的软件项目，还是小规模的 MIS（Management Information System，管理信息系统），针对这些部分的软件功能总是被客户认为是最有价值的。

（2）对于金融保险行业来说，由于保险公司的基本职责是分摊风险和补偿损失，所以，管理保单和保险人信息的业务系统、单证系统、评估风险的定损系统等就是非常有价值的软件系统。

（3）对于教育培训行业来说，因为学校的核心职能是教书育人，因此，与教研、教学、考试、评价等业务相关的软件系统，以及支持上述业务开展的教育资源库软件、课件制作工具、电子图书馆软件、教学科研支撑系统等就是高价值的软件系统。

（4）对于互联网的网民来说，面向信息交换和传递的电子邮件系统、面向查找信息的搜索引擎、保证系统安全的防火墙和杀毒软件、面向网络沟通和交流的即时通信工具（例如，MSN Messenager、QQ）等就是高价值的软件系统。

总之，选择和确定软件项目，必须首先考察软件应用的行业、业务和目标，以便判明要建设的软件项目价值。

2．评估项目约束、风险、收益和代价

在判断出一个具有潜在价值的软件项目后，还应评估项目实施的约束、风险、收益和代价。通常这部分内容可以在项目的可行性分析阶段完成。

所谓项目约束，是指在产品开发过程中，不能做什么的原则。这些约束有些来自客户，有些来自企业本身，有些来自外部环境。可能包括：

（1）企业约束。例如，项目是否符合企业定位、商业价值观、经营方式，是否是企业需要优先实施的项目等。

（2）资源约束。例如，时间、人力、资金、设备等各种资源是否能承担并满足投入计划等。

（3）能力约束。例如，项目技术难度、企业技术能力、人员素质和水平、项目开发周期等。

（4）环境约束。例如，项目是否符合行业标准、是否符合国家政策规定、是否符合当前的社会环境、是否符合行业发展水平等。

（5）用户约束。例如，使用软件的用户素质和认识水平、用户业务的限制、用户的工作方式和行为习惯等。

一些明显的约束条件可以在立项阶段明确评估出来，隐性的约束则容易被软件开发

者忽视（例如，企业资源投入的变化、国家政策变化等），从而导致各种项目执行中的风险。项目约束通常是开发者不可控制的因素，对于这些约束，必须时刻关注方能尽可能规避风险，如果明显违背这些约束条件，就会导致软件项目不可避免的失败。

对于购买产品或技术服务的客户来说，除了考察上述项目约束外，还应该评估项目实施后的影响。例如，对自身业务变更的影响、组织机构和人员职责的影响、相关的系统维护、运行规约和规章制度等，以及项目的效益、当前成本、未来的总持有成本（Total Owner Cost，TOC）是否能接受等。

评估项目约束、风险、预期收益和代价后，即可筛选掉多数不符合企业要求的建议项目。

3．评估项目的多种实施方式

对于已经确认有价值，并且有能力开发的软件项目，则可以进一步参照企业现状考察项目的实施方式。这种实施方式通常既包括前面对项目风险、预期收益和资源开销的评估，也包含企业对现阶段经营目标和现有资源如何合理运用的考虑。这个过程通常由项目的负责人和企业中高层经理进行决策，决策结果决定项目的实施优先级和具体的实施方式。

企业完成软件项目的具体方式并不单纯限制于自己组建开发团队进行项目开发的策略。根据具体情况不同，还可能使用诸如转包开发业务给外部公司，直接 OEM（Original Equipment Manufacture，原始设备制造商）软件产品并进行系统集成，购买关键技术并进行软件集成方式的开发，或者企业自己完成技术方案和设计，然后，寻求外部公司进行代工等方式。对这些项目实施方式的取舍，其主要依据依然是对项目风险、收益和资源开销的平衡的考虑，其根本目标是为了优化和合理运用投入项目的资源。

由于企业实施软件项目，除技术、管理、应用的角度外，隐含有企业经营和加强竞争力目标的判断，例如，成本领先策略（通过合理选择实施方式缩减成本等、差异化策略（建立和加强与合作伙伴的关系）、专注化策略（提高效率和降低项目风险并专注在自己擅长的领域），因此，评估项目的多种实施方式就显得特别重要，如图 1-2 所示。

4．平衡地选择适合的方案

人们在选择可行的方案时，总是希望能尽量得到一种高质量、低成本的产品和方案。软件开发人员通常也很愿意在产品开发中，向产品中加入激动人心的“创造性”的内容。另一方面，客户单位在面对诸多的投标方案时，会听到各种各样关于技术先进性、快速开发、产品质量稳定可靠、价格如何低廉、推荐的方案有多少成功应用等宣传。这些内容本身存在很多矛盾，简要列举案例如下：

（1）技术风险。采用成熟的技术则可能享受不到新技术带来的好处，但流行的新技术可能是易变化的，从而导致风险。新技术也意味着技术开发者需要更多的学习时间，从而导致开发成本的增加。

（2）用户锁定性。不基于某种快速开发技术或平台构造的产品可能会提高项目开发

时间而导致更多的开销和成本，但基于某种平台的产品又可能使得用户未来“绑定”在某种平台之上，减少了自由选择能力，甚至未来被迫接收厂商的定价和服务。

（3）扩展性。不考虑系统的扩展性，将导致业务变更时受阻于已经建成的 IT 设施，重新改造这些 IT 设施既增加成本又导致大的影响，几乎是一种灾难。另一方面，如果过多考虑系统的扩展性，用户常常可能在当前的采购中购买了一些自己并不需要的特性，从而支付了更多的成本，由于 IT 技术发展迅速，当用户期望进行系统升级的时候，常常会发现原来的开发平台早已被淘汰和抛弃。

（4）目标偏离。出于希望达成交易的需要，供应商更愿意宣传技术先进、价格低廉、快速提交、软件的新特色和新性能等因素。在用户对 IT 技术、作用和代价不甚明确的时候，容易受到宣传的影响从而偏离对原有目标的关注。

事实上，对软件功能和性能的要求常常是充满矛盾的，任何时候都从不存在一个完美无缺的方案，只存在一个对当前的项目目标相对比较适合的方案。选择项目的基本立场，应该是“适合”，而不是尽可能的“好”。实际上，任何超出预期设定目标的“好”性能，通常都意味着某方面更多的成本或者潜在的风险。因此，平衡地选择适当的项目，是系统项目选择非常重要的原则之一。

希赛教育专家提示：所谓适合的方案，就是平衡考虑开发单位利益和客户满意度的方案。

图 1-4 是 Noriaki Kano 博士创造的顾客满意度的质量模型图。

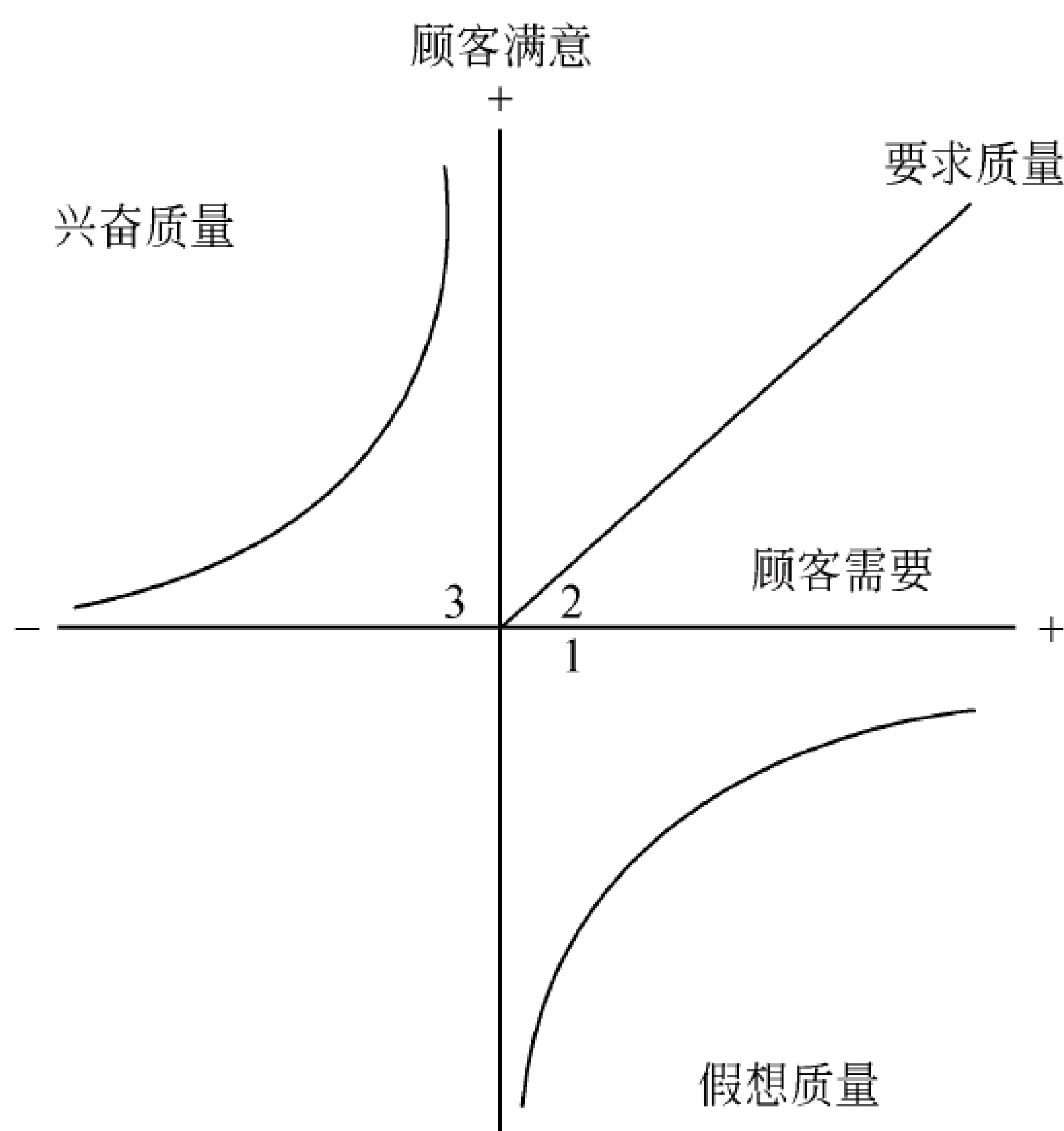


图 1-4　顾客满意度的质量模型

（1）要求质量：客户认为产品应该做到的功能或性能，实现越多客户会越满意。

（2）假想质量：客户想当然认为产品应具备的功能或性能，客户并不能正确描述自

己想要得到的这些功能或性能需求。软件工程手段的需求获取、需求管理和原型化方法并不能有效获取涉及假想质量的软件需求。

（3）兴奋质量：客户要求范围外的功能或性能（但通常是软件开发人员很乐意赋予产品的技术特性），实现这些性能客户会更高兴，但不实现也不影响其购买的决策。兴奋质量是控制在开发人员手中的，开发人员可以选择实现更多的兴奋质量，以便得到高满意、高忠诚度的客户；也可以（出于成本或项目周期的考虑）选择不实现任何兴奋质量的功能（例如，为 MIS 软件制作漂亮的界面）。

显然，项目开发方更多考虑的是项目风险和回报，而客户更多关心的是成本和购买后的满意度，一个好的方案必须平衡考虑这些因素。系统分析师应尽可能用技术手段来平衡这些彼此对立的要求，保证在项目预期投入资源可接受的范围内，尽量实现客户要求质量对应的功能和性能，发掘客户假想质量对应的功能要求并进行沟通、确认，但按自身所服务企业的经营目标平衡考虑客户兴奋质量的实现策略（是努力提供兴奋质量的功能，争取忠诚的客户获得远期潜在的收益，还是削减这些功能以便最小化项目的成本）。

系统分析师常犯的一个错误，就是用自己对技术的兴趣产生的兴奋质量，来替换客户最基本的要求质量和假想质量；而企业经营者常犯的错误，则可能是对客户提出的合理要求质量内容麻木不仁，或者走向另一个极端，不加区分地把一切未经评估的假想要求质量不断指派给软件开发团队。这些都是错误的做法。

1.1.4　项目提出和选择的结果

信息系统项目提出和选择的结果，最终会以产品/项目建议书的方式来体现。典型的应用场景是：

（1）在投标项目中，产品/项目建议书通常是乙方提交给甲方竞标方案的一部分。

（2）企业在确立了要开发某类型产品后，对该产品进行多角度的评估，最终项目立项人向上级提交供决策的建议报告的主要内容就是产品/项目建议书。

产品/项目建议书是一个包含多种综合内容的报告，涉及的范围通常要比 GB/T 8567——2006 标准中规定的《项目可行性研究报告》（请参考第 10.2 节）的内容要更全面。在项目建议书中，可能包括以下内容：

（1）用户单位、项目或产品的立项背景、需求来源和目标性的介绍。

（2）用户的内外部环境、组织机构、现有的 IT 设施情况等。

（3）用户的业务模型和业务规划。

（4）预期要建设的技术系统在用户业务中的位置和作用。

（5）信息化后的用户业务模型、软件应用方式、部署环境、运行规则、管理规范等。

（6）为实现信息化业务模型，技术系统的产品需求定义（功能、性能、约束）和部署方式。

（7）产品或项目的技术框架。

（8）项目的要点、技术难点、主要实施障碍等。

（9）项目或产品的可行性研究结果。

（10）项目可选择的实施方式、组织方式、沟通和协调机制等。

（11）项目的资源范围和预算（包括人、财、物、时间等，请参考第 11 章）。

（12）项目的成本/收益分析。

其他项目建议书可能包含的内容，或以单独文档列举的内容可能包括以下内容：

（1）项目风险及影响评估。

（2）项目进度计划。

（3）项目质量计划。

（4）项目过渡期资金的获得方式、财务计划。

（5）产品或项目的商务模式、盈利模式论述。

（6）同类产品或公司的市场调查结果，以及竞争性比较。

（7）企业成功案例、资质等。

（8）商务条款或供应商/客户合同。

项目建议书标志着项目立项和选择阶段性工作的完成，一旦项目建议书被批准通过，项目即可进入正式的开发准备和实施阶段。

1.2 定义问题与归结模型

对于经过系统计划、选择并确定要实施的项目，定义问题与归结模型过程的主要目的是得到系统开发清晰的目标、功能清单、性能要求（包括约束），以便确定系统目标，为后续的工作进行准备。这个阶段不解答的问题是软件系统架构、功能划分、实现算法这样的设计问题。

1.2.1 方法论模型

图 1-5 描述了问题定义与归结模型的方法论模型。问题定义和归结模型位于系统项目立项和确立之后的阶段。

1．自下而上的软件工程技术进展

传统软件工程方法的工作范围，在定义了软件概念模型后即进入软件的概要设计阶段。软件设计的重点被放在数据结构和算法的选择上，对于大规模的复杂软件系统来说，对总体的系统结构设计和规格说明比起对计算的算法和数据结构的选择明显重要得多。传统软件工程方法显得力不从心。

在系统分析和设计层面上，现代软件系统设计方法将软件架构设计独立出来。软件架构的“4+1”视图从 5 个不同的视角包括逻辑视图、进程视图、物理视图、开发视图和场景视图的角度对软件系统作了高阶抽象，揭示了系统需求和构成系统的元素之间的

对应关系，并提供了进一步设计决策的基本规则，试图架设从软件分析设计手段到软件需求的桥梁，如图 1-6 所示。

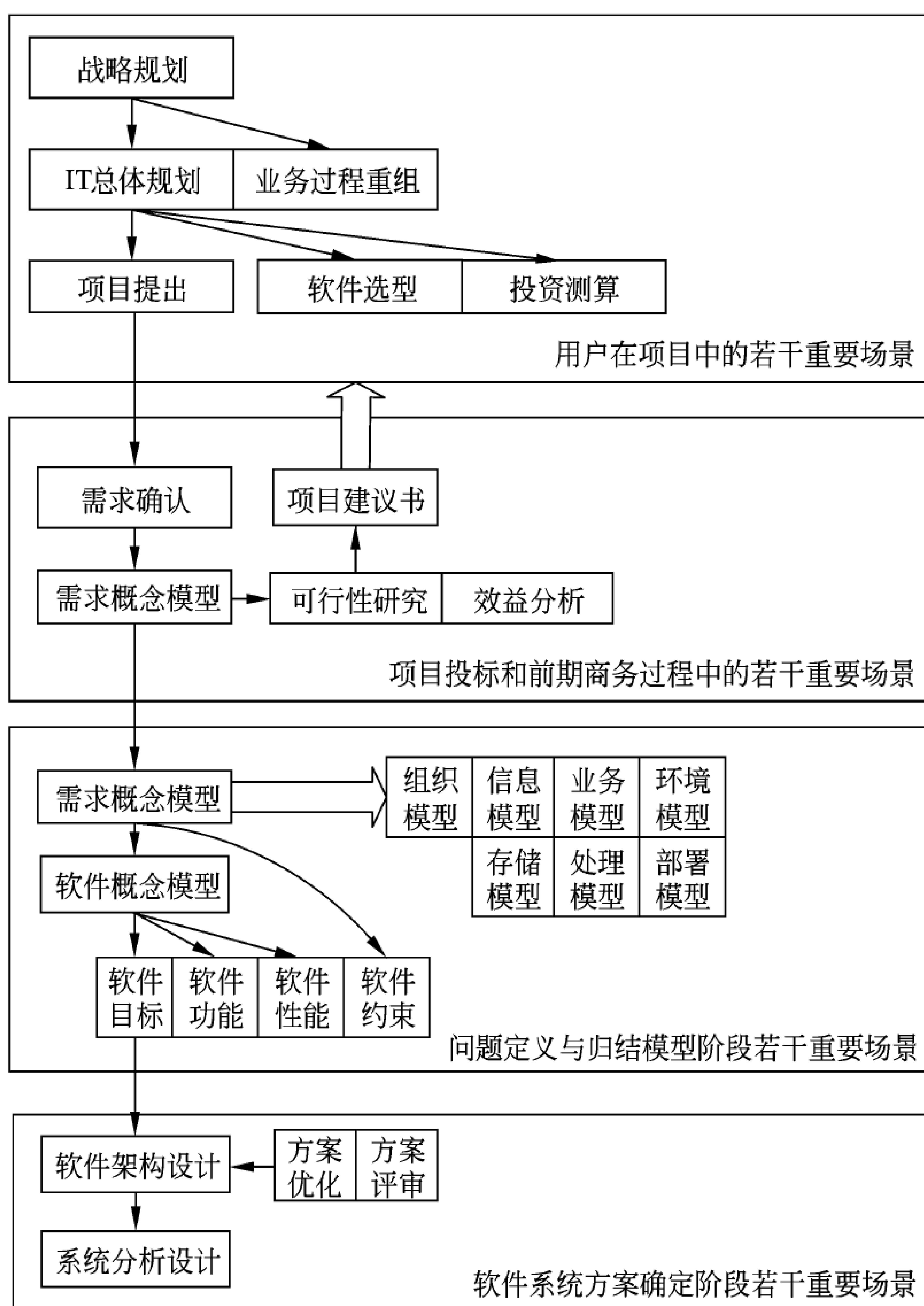


图 1-5　系统设计的归结模型

关于软件架构的详细知识，请参考本套丛书中的《系统分析师技术指南（2009 版）》的第 11 章。

2．自上而下的需求分析技术进展

软件概念模型分析出的软件需求，实际上包含了软件功能需求和软件性能需求（软件约束是从“不能做什么”的角度来定义软件需求，因此，也可归并到性能需求中）。在

需求分析方面，人们进一步扩展了用于描述软件需求的其他模型。通常包括：

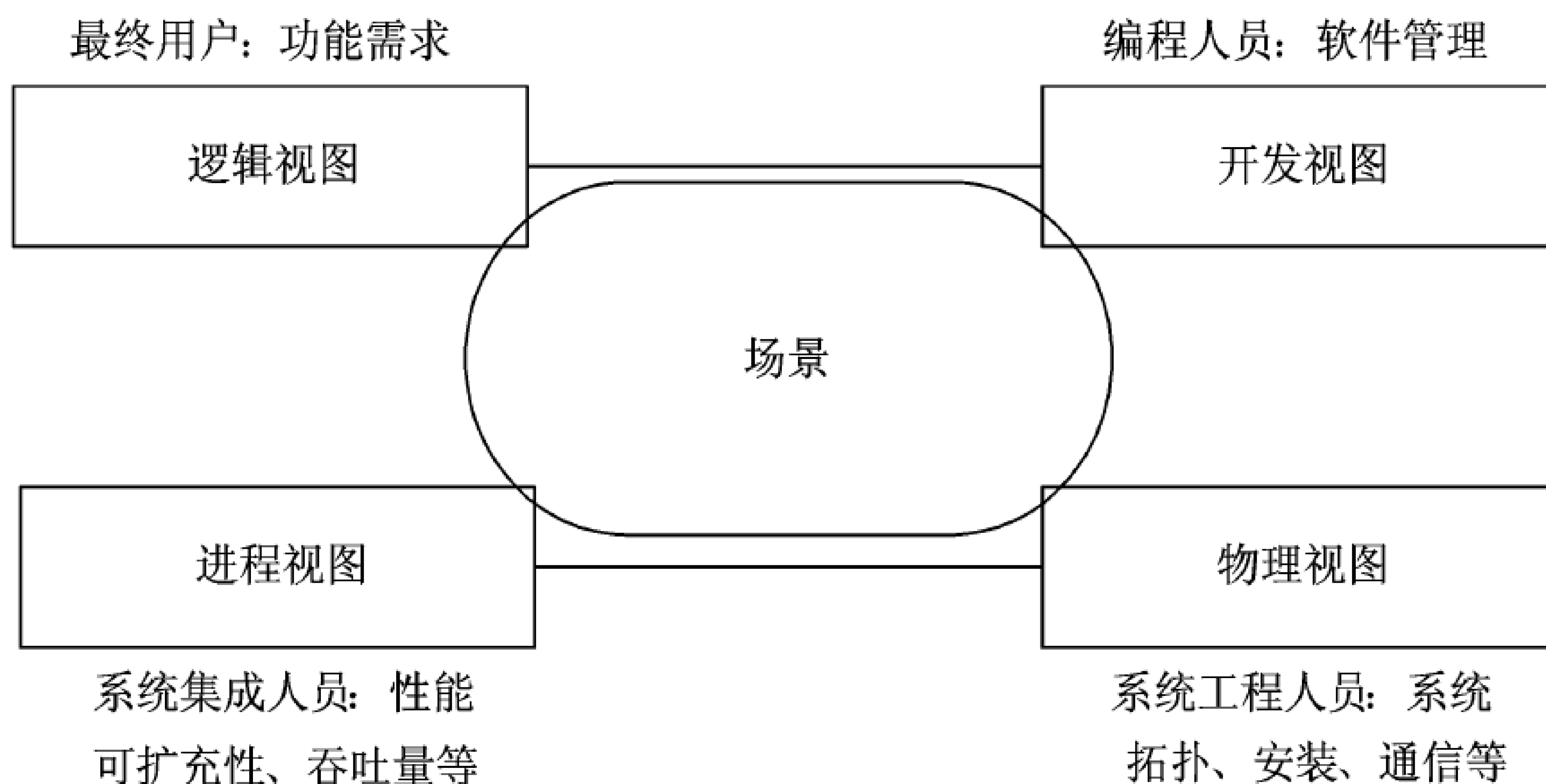


图 1-6 软件架构的“4+1”视图模型

（1）组织结构和人员模型。研究用户组织结构和人员模型，以及软件功能在用户组织结构上的分配和权限、软件对用户组织和人员的影响等。

（2）信息模型。研究软件系统涉及到的数据、信息、文件以及未来的存储、加工等需求。E-R（Entity-Relationship，实体-联系）图、DFD（Data Flow Diagram，数据流图）、数据字典等均可用于信息模型的建模。

（3）业务模型。研究项目或产品应用后，信息化的业务模型、信息化后的业务处理方式和业务流程、人机之间的接口和协作关系等。从业务模型出发，不仅可以得到工作流程，而且还可以得到大多数未来软件系统的调度规约、运行规约和相关的维护、管理制度。高阶的业务模型图可以是很简单的业务流程概念图，而细化的业务模型图可详细分析具体业务的工序甚至工步。

（4）环境模型。描述系统的部署方式、安装方式、环境要求等。

这些模型解释了系统概念模型（系统应该做什么）之外更广泛的问题，严格的项目可使用 UML（Unified Modeling Language，统一建模语言）进行描述，实际工作也常使用 Visio 图、Word 绘图进行描述。

1.2.2 实现步骤

虽然并不存在一个可套用的问题定义和归结模型的模式，但通常下列工作总需要在问题定义阶段完成：

（1）用户需求细节的获取、采集、整理、确认、分析和管理，可能会用到调研、需求工程、原型化方法、需求管理等手段。

（2）就原有的用户业务建模，包括企业级、系统级和关键业务的业务模型。并依据模型分析的结果，参照已经获取到用户软件需求进行匹配和确认，确认业务模型中的关

键域和需要解决的重点问题。

（3）假想信息化后的业务模型图，建立分层的、高阶的其他扩展项目视图（组织模型、信息模型、业务模型、环境模型），包括这些模型的优化和改进。在这些模型中比较、分配和确认用户需求。这些模型的总和，形成了用户的需求概念模型。

（4）根据需求概念模型使用系统分析、建模手段建立概念模型，并将其进一步细化为系统的全部目标，包括功能清单、性能清单、约束条件。

（5）将功能、性能等重新分配到需求概念模型，可以得到未来项目或产品构成与实施的基本要素和原则。

（6）依据概念模型和功能点清单、性能清单进一步展开后续的架构设计和软件设计工作。

在这个过程中，用户描述的需求经过了两步映射被转换到可进行系统设计的需求。第一步是试图将模糊的用户需求逐步确认并建立需求的概念模型，然后以这个模型为依据，得到各种细致的需求与规划模型；第二步是确立系统概念模型，并从概念模型出发，确定系统需要实现的所有功能、性能要求和约束条件。这些需求才是开发人员能够具体去设计和实现细节的基础。例如，需求概念模型中定义的要求“应该尽量降低用户的操作和使用难度”，可能被细化为概念模型中的“使用简洁的菜单和窗口”、“采用助输入手段”、“实现在线帮助和提示”等具体的功能要求。

系统需求最终被归结为一系列清晰的系统功能描述清单和系统性能清单。功能描述了系统必须完成的任务，性能描述了为了实现主体功能正常使用必须达到的性能指标，约束描述了主体在具体的环境和场景下的不能作什么的限制条件，多数也可以归结到对系统性能的需求。

对于分析出的系统功能要求，可根据重要性、优先级、难度和对客户的价值度进行再次评估和排列。它们是未来被管理的系统需求和开发边界的基本依据。对于分析出的性能要求，同样需要进行重要性和优先级的排列。其中性能的选择，可以参照一定的指标选取原则，根据项目质量评估的标准化体系进行取舍。

1.2.3　典型方法

定义问题和归结模型阶段使用到的典型方法主要有需求工程、领域建模和业务过程建模。

1．需求工程

需求工程用于获取、整理、加工、确认和管理需求。需求工程的目的是通过与用户广泛地交流，确定应用系统的目标。以工程化的方法来提出、分析、验证、组织和管理需求的过程，同时它也鼓励用户以一种积极的方式参与到需求分析活动中，通过在整个系统生命周期中，用户参与和领域专家的指导作用，促使目标系统最大地满足用户需求。

需求工程是一个不断反复的需求定义、记录和演化的过程，并在最终达到需求的稳定。大致可以分为5个阶段：

（1）需求形成。这个阶段主要是处在一个开发者和用户不断的交流和讨论，使得要开发的系统逐渐清晰。通过这个过程，最终使得整体想法完全成型。这一阶段的需求形态是比较模糊的和破碎的，大多数需求还只存在于客户和开发组织的人员的头脑里面。

（2）需求获取。通过积极与用户交流，捕捉、分析和修订用户对目标系统的需求，并提炼出符合问题解决领域的用户需求。将这些明确的需求使用标准的方式，无矛盾和歧义地记录下来，从而成为系统蓝图完整描述的一部分。需求的记录可以通过需求管理软件进行管理，也可以是一个简单、有层次的结构化描述文本。这一阶段的需求形态已经独立于客户或者开发组织内部的个人而存在，成为可以交换和存档的文件。因此，遵循必要的规范和采用标准描述方式是很重要的。

（3）需求建模和规格说明。根据需求分析，对已获取的需求进行抽象描述，为目标系统建立一个概念模型。同时对需求模型进行精确的、形式化的描述，为计算机系统的实现提供基础。

（4）需求验证。以需求规格说明书为依据，使用系统功能和项目约束进行需求验证、保证这些需求和约束都能得到满足，同时也可以使用模拟或快速原型方法，验证需求规格说明的正确性和可行性。

（5）需求管理。跟踪和管理需求变化，支持系统的需求演化，实现需求的双向跟踪。

希赛教育专家提示：需求工程可分为需求开发和需求管理两大工作，其中需求开发分为需求获取、需求分析、编写规格说明书（需求定义）和需求验证4个阶段，需求管理包括定义需求基线、处理需求变更及需求跟踪等方面的工作。这两个方面是相辅相成的，需求开发是主线，是目标；需求管理是支持，是保障。有关这方面的详细知识，请阅读本套丛书中的《系统分析师考试全程指导（2009版）》的第9.6节。

2．领域建模和业务过程建模

传统软件工程在采集需求和分析需求时，通常并不考虑需求根源和变化的由来，也没有对研究系统功能在用户的整个业务执行中的作用和协作关系。这样，系统需求的质量以及这些需求在多大程度上能与用户的最终目标相匹配，就比较依赖于系统分析师、用户代表和领域专家的个人水平和沟通能力。对于很多对自身业务理解或IT技术不充分的用户来说，即使开发公司具有充分的技术积累，不能得到充分准确的需求也会严重影响后期的系统质量和大大增加项目风险。这些问题需要依赖对领域和对业务过程进行建模来解决，图1-7是一个业务过程建模的例子。

用于领域建模和业务过程建模的工具和方法学，与系统分析设计使用的工具和方法学是一致的（例如，使用UML对高阶的商业过程进行建模），只是针对的领域不同而已。目前，在领域建模和业务过程建模领域，尚没有统一的标准，但已经存在一些共识和有益的探索。

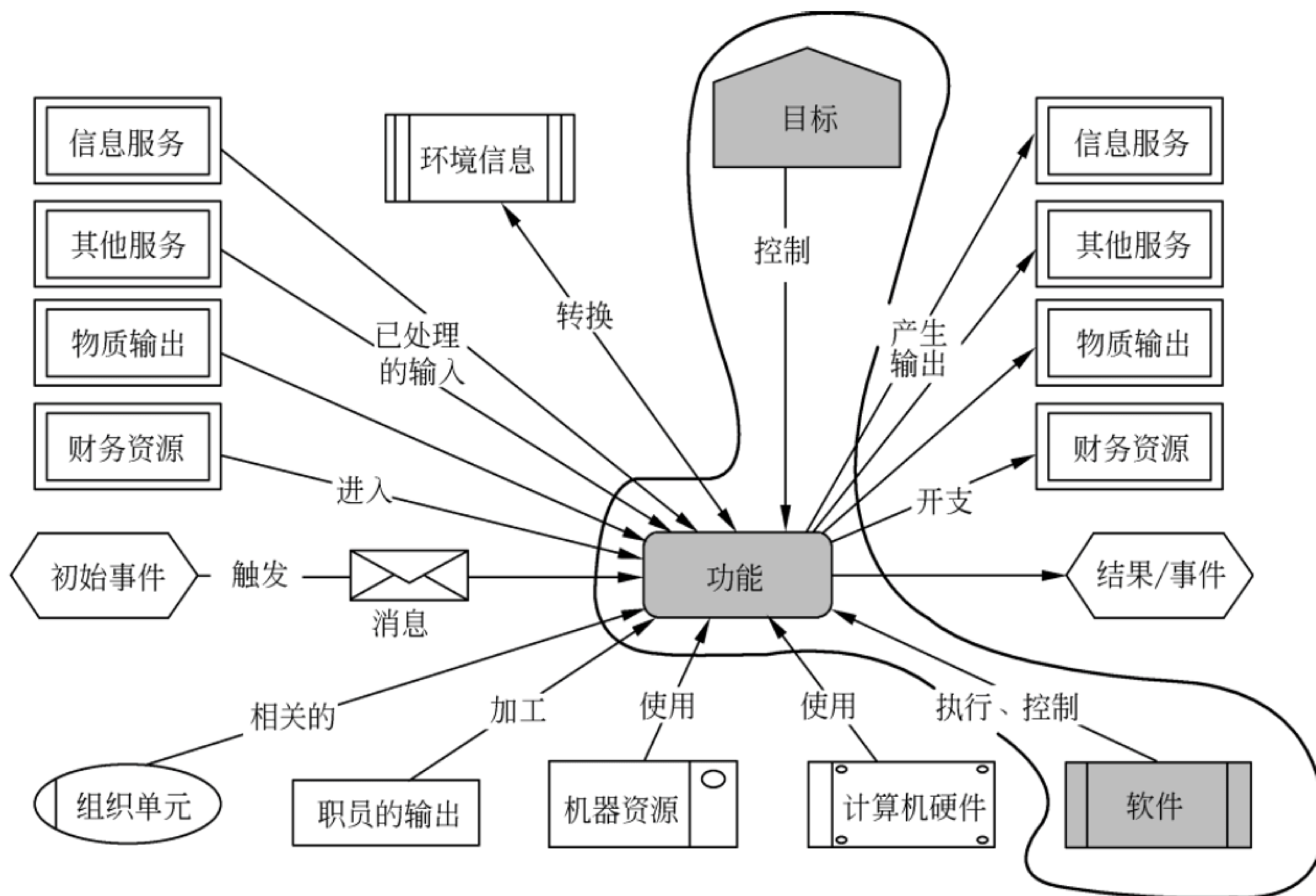


图 1-7　业务过程建模的示例

由于需求工程的输入是初步的需求定义，输出是文档化的需求定义；软件工程的输入是文档化的需求定义，输出是软件。领域工程、需求工程、软件工程三者之间本身存在衔接和配合关系。对于质量、效率和目标的共同需求，导致对领域建模、需求建模和软件设计建模进行一体化建模的呼声不断出现，而软件工程领域的激进实践者，已经在试验模型驱动的方法，包括模型驱动编程（Model Driven Programming，MDP）、模型驱动架构（Model Driven Architecture，MDA），以及模型驱动需求定义（Model Driven Requirements，MDR）等。就某一个系统的开发而言，是否使用这些流行的方法并无定论，但总的原则在于，需要高度重视业务和需求。

希赛教育专家提示：不管使用哪一种方法进行分析与设计，越接近问题的本质，就越有助于问题的解决。

1.3 可行性研究

可行性研究的主要目的就是回答一个问题，即所提出的项目是否可以完成。这就要求系统分析师能够进行一次压缩和简化的系统分析与设计，也就是在较高的抽象层面上进行分析和设计。不过，需要注意的是，可行性研究毕竟不是解决问题，而是研究问题的范围，探索这个问题是不是值得去解决，根据现有的情况是否有能力，是否有可能找到较好的、成本效益合算的解决方案。

1.3.1 可行性研究的意义

在项目计划和选择的过程中，需要完成的首要事情是对项目进行估算。项目估算的范围涉及方方面面，例如，项目或产品开发的范围、投入和回报、项目风险、作用和意义等。在传统的信息系统建设方法中，是以可行性研究的方式来组织对项目的主要估算内容的。在企业实际的业务过程中，可行性研究通常作为一个重要的环节，被包含在整个项目立项、或项目选择和确认的过程中。

可行性研究的范围可能覆盖很广泛的技术、经济、执行、环境等各种需要评估的因素，但它并不是最后的精细计划（例如，项目的时间进度以及人员安排）。通常，在进行可行性研究的阶段，甚至项目的目标或产品的最终方向也是高度易变化的。但可行性研究的意义在于，虽然可行性研究不能指出项目最终的精细计划和方向，但可行性研究可以在项目定义阶段用较小的代价识别出错误构思的系统，从而规避未来更多的资源投入的损失（时间、资金、人力、机会），或者因遭遇到无法逾越的技术障碍或环境障碍导致的不可避免的失败。

对于那些可行性研究表明可执行的项目来说，可行性研究的结果也不承诺系统的收益一定很巨大，或技术风险和资源投入就一定很低，但可行性研究的结果设立了一个"底线"，即"如何做，则风险和收益是什么这样"的控制范围。这些评估结果给了未来的项目评估、项目风险控制甚至在资源剧烈变化的情况下，有计划、有重点地削减功能，重定义项目开发范围，或者选择项目实施的方式提供了非常有价值的方向性指引。

1.3.2 可行性研究的内容

在传统的系统工程理论中，可行性研究涉及的主要方面包括经济可行性、技术可行性、法律可行性、执行可行性和可选择性。

1．经济可行性

经济可行性主要评估项目的开发成本，以及项目成功后的可能经济收益。多数项目只有开发成本能控制在企业可接受的总数量和计划内的时候，项目才有可能被批准执行。而经济收益的考虑则非常广泛，例如，项目技术开发的直接现金收入、新产品在生命周期中预期的总销售收入、技术积累、对公司业务和产品线的完善和支持、开辟新市场和利润增长点、进入预期能带来较高收益的新市场、提高客户满意度和忠诚度、打击竞争对手抢夺市场份额、获得新的信息化能力从而改善经营或管理格局等。

2．技术可行性

技术可行性评估对于假想的系统需要实现的功能和性能，以及技术能力约束。技术可行性分析可通过提问-回答的方式来进行论证，包括技术、资源和目标等。

（1）技术：现有的技术能力和 IT 技术的发展现状是否足以支持想象中的系统目标实现。

（2）资源：现有的资源（掌握技术的职员、公司的技术积累、构件库、软硬件条件等）是否足以支持项目实施，技术风险是否在评估的范围内。

（3）目标：在目前设定的系统目标中，哪些目标会遭遇到较强的技术障碍，尤其是那些被设定为必须实现的系统目标。

由于在可行性研究阶段，项目的目标是比较模糊的，因此，技术可行性最好与项目功能、性能和约束的定义同时进行。在可行性研究阶段，调整开发目标和选择可行的技术体系均是可用的手段，而一旦项目进入开发阶段，任何方向性的调整都意味着更多的成本、代价和风险。

需要指出的是，技术可行性研究绝不仅仅是论证在技术上是否可实现，实际上包含了在当前资源条件下的技术可行性，以及对技术手段的优化选择。前者要求系统分析师充分考虑项目约束，后者要求系统分析师进一步考虑技术手段的优化和对企业经营目标和项目根本价值的支持。

投资不足、时间不足、预设的开发目标技术难度过大、没有足够的技术积累、没有熟练的职员可用、没有足够的合作公司和外包资源积累等均是典型的技术可行性的约束。只具有技术背景的系统分析师很容易只考虑技术手段是否能实现，忽视了当前的企业定位、资源条件和环境，从而对技术可行性研究得出过于乐观的结果，这种错误判断对后期的项目实施会导致灾难性后果。

不关心项目技术手段优化的系统分析师，只能有效地“完成”项目和产品的开发，不能进一步通过控制成本、优化技术手段等来提升项目价值。

希赛教育专家提示：加强前期的项目调研，寻求专家的咨询，以及采用具有大量成功应用案例、被广泛支持的技术标准等均有助于改善项目的技术可行性。

3．法律可行性

法律可行性评估可能由系统开发引发的侵权或法律责任，可能包括合同的订立和条款、职责、侵权情况的设定、违约、争议解决等方方面面的内容。

法律可行性还包括国家政策和法律的限制，例如，在政府信息化的领域中使用未被认可的加密算法、未经许可在产品中使用了其他公司被保护的软件技术或构件等。

4．执行可行性

执行可行性主要评估预期的系统在真实环境中能够被应用的程度和实施障碍。例如，ERP 系统建成后的数据采集和数据质量问题，或客户工作人员没有足够的 IT 技能等。这些问题虽然与系统本身无关，但如果不经评估，很可能会导致投入巨资建成的系统毫无用处。

执行可行性还需要评估对用户的各种影响，包括对现有 IT 设施的影响、对用户组织机构的影响、对现有业务流程的影响、对地点的影响、对经费开支的影响等。如果某项影响会过多改变客户的现状，需要将这些因素作进一步的讨论，并和系统的使用者沟通，提出建议的解决方法。对于特别复杂的系统，常要求用户单位在实施项目前，进行战略

规划和业务过程重组（Business Process Reengineering，BPR），充分评估信息化的影响。

5．可选择性

评估系统或产品开发的其他可选方法，例如，系统构架方案（即采用哪种架构，为什么采用这种架构），网络接入方案（在众多的网络接入方法中进行比较分析，采用哪种方法更经济且符合系统需要）。系统分析师需要在各种方案中给出优先级别的排列。

1.3.3 可行性研究的步骤

在清楚了可行性研究的意义和内容之后，本节就可行性研究工作的步骤做一个总结性阐述。

1．核实问题定义与目标

系统分析师开始正式进行可行性研究工作之前，首先要做的一个工作，就是对该项工作的基础-问题定义-再次核实。具体来说，就是仔细阅读问题定义的相关材料，对该问题所涉及的领域知识进行学习、考证，然后通过走访相关人员进行验证与核实。

这一步骤的关键目标是：使得问题定义更加清晰、明确、没有歧义性，并且对于系统的目标、规模以及相关约束与限制条件做出更加细致的定义，确保可行性研究小组的所有成员达成共识。

2．分析现有系统

对现有系统的仔细分析与研究是十分重要的一项工作，因为它是新系统开发的最好参照物，对其充分分解有助于新系统的开发。这么说的原因，主要是基于以下几点考虑：

（1）现有系统已实现的功能通常也是新系统要实现功能的一部分。

（2）新系统一定是在现有系统基础上的升华，毕竟如果旧系统没有问题，就不会有新系统开发的需求。

（3）现有系统的运行成本分析的结果是衡量新系统的经济可行性的重要参照物。

希赛教育专家提示：从字面上的理解会容易产生一个常见的误区，就是认为现有系统一定是计算机系统。其实，这里的“现有系统”不仅包括旧的计算机系统，还包括非计算机系统（手工流程）。纯手工运作的流程也可以看作是一个系统，对于这样的系统的研究，可以借助于业务流程图，了解原先的运作模式，从中找出问题的瓶颈，这也对新系统开发有着很大的辅助作用。

有关现有系统的分析方法，将在第 1.4 节进行详述。

3．为新系统建模

在问题定义、现有系统研究的基础上，就可以开始对新的系统进行建模，而建模的目的则是为了获得一个对新系统的框架认识和概念性认识。通常可以采用以下几种技术。

（1）系统上下文关系范围图：其实也就是 DFD 的 0 层图，将系统与外界实体（可能是人、可能是外部系统）的关系（主要是数据流和控制流）体现出来，从而清晰地界定出系统的范围，实现共识。

（2）E-R 图：这是系统的数据模型，这个阶段并不需要生成完整的 E-R 图，而是找到主要的实体以及实体之间的关系即可。

（3）用例模型：这是系统的一个动态模型，以角色（actor）和用例（use-case）整理出系统的主要功能框架，这个阶段应该大部分都处于概念级，每个用例也无需花太多的时间确定细节，只要能够勾画出系统的雏形即可。有关用例模型的详细知识，请阅读第 2.5.2 节。

（4）域模型：这是采用面向对象的思想，对于系统中主要的实体类找到，并说明实体类的主要特征和它们之间的关系。

（5）IPO（Input/Process/Output）表：这是采用传统的结构化思想，从输入、处理、输出的角度进行系统描述。

希赛教育专家提示：这个阶段的所有模型都是不够精确的，而只是一个框架，要从宏观的角度把握系统。否则，将陷入无休止的细节化工作。

4．客户复核

可以这么说，在第 3 步中建立起来的系统模型是系统分析师眼里的问题定义，不一定能与客户心目中的目标一致。因此，完成系统模型的建立之后，一项十分重要的工作就是与客户一起进行复核。当然，这个时候模型将成为讨论和分析的基础，所以，使用客户更容易接受的模型将显得十分重要。

如果在这个过程中，发现模型与客户的目标有不一致的地方，就应该再次通过访谈、现场观摩、对现有系统分析等手段进行了解。然后，在此基础上修改模型。因此，也可以说，1～4 的步骤是一个迭代的过程，周而复始，直至客户确认了新的系统模型为止。

5．提出并评价解决方案

前面的工作还是停留在“系统解决什么问题”上，只是更加清晰地定义和说明清楚。当客户与系统分析师对要解决的问题有了一个清晰的共识之后，接下来的工作就是提出解决方案，这也是系统分析师很重要的工作之一。

在这个阶段，应该尽量列举出各种可行的解决方案，并且对这些解决方案的优点、缺点做一个综合性的评价，以便于下一步决策。不过要注意的是，对于那些明显不可行的（例如，技术上还没有相应的办法、经济角度明显不可行的、违背企业或行业实际情况）解决方案，应该直接过滤掉。

6．确定最终推荐的解决方案

在这个阶段，系统分析师需要明确地指出项目是否可行，如果可行的话，什么方案是最优的。对于这两个问题的回答，是可行性分析研究工作的核心目标。因此，在各种解决方案提出之后，紧接下来就应该从中选出一个相对来说最合理、最可行的解决方案，更加详细地说明理由，并且还要对其进行更加完善的成本/效益分析。

7．草拟开发计划

在上一个阶段，系统分析师对主要推荐的解决方案进行了详细的成本效益分析，对

工作内容和工作量都有了一个详细的了解。接下来，就需要制订一个最粗略的开发计划，说明开发所需的资源、人员和时间进度安排。这也将做为可行性分析的一个重要依据，和立项开发后制订项目管理计划的基础。

8．提交可行性研究报告

最后，就是将可行性研究的结果整理成文，形成可行性研究报告，提交客户和管理层，进行审查通过。在这个阶段，系统分析师还应该制作一些相应的讲义，为客户和管理层做介绍和说明。有关可行性研究报告的详细内容，请阅读第 10.2 节。

1.3.4　成本效益分析

具体来说，成本效益分析可以分成两个部分的内容，一个是成本估算，另一个是效益分析。中小规模项目的成本效益分析可包含在可行性研究的经济可行性分析报告中，而较大规模的项目，需要在项目计划书中单列文本进行成本效益的分析。

1．成本估算

对于软件系统而言，主要的成本是人力资源成本，另外还包括一些直接的成本、设备采购的成本等。相对而言，硬件设备等其他的一些相关成本的评估会比较容易一些，最难的就是人力资源的成本分析。因此，对系统工作量（用人月、人年等单位进行说明）进行合理、科学的评估，并在此基础上进行计算是很必要的。

要想准确地估算出工作量，通常可以借助的工具是历史数据和经验模型。历史数据就是指开发团队以往从事项目的情况，可以通过以类似系统项目做参照的方法，这种方法也称为类比估算法；而经验模型则是一些软件工作量估算方面的研究总结，常用的有功能点分析、COCOMO 模型等。

具体来说，可以分成以下几步进行：首先，进行工作任务分解，将目标细化；然后，就每一个工作任务包，与具体的开发团队进行共同分析，使用功能点分析法或其他相关的分析法进行估算，并且将估算的结果与类似项目的历史数据进行比较，做出微调；最后，使用 COCOMO 模型等，计算出相应的人月数。在这个过程中，还应该根据类似项目的历史数据推出项目团队的平均产能，从而使得估算值更有代表性。

这个阶段的工作，对于系统分析师而言，很重要的知识技能在于软件度量与估算方法、技术的掌握方面。

希赛教育专家提示：在这个阶段，是不可能得出精确的估算值的，这个观念必须让开发团队、管理层和客户很清晰地认识到。否则，即使违心地给出所谓“精确”的估算值，也显得没有任何意义。

2．项目可能涉及的成本

项目的成本部分，通常可能包括基础建设支出、一次性支出和运行维护成本等。

（1）基础建设支出。例如，房屋和设施、办公设备、平台软件、必须的工具软件等购置成本。

（2）一次性支出。例如，研究咨询成本、调研费、管理成本、培训费、差旅费等，以及其他一次性杂费。

（3）运行维护成本。例如，设备租金和定期维护成本、定期消耗品支出、通信费、人员工资与奖金、房屋租金、公共设施维护等，以及其他经常性的支出项目。

除购买产品或服务的开支外，通常还包括各种系统维护、改进、培训、招聘新职员、变更业务流程等各种应用方面的开销。虽然这些成本可能不会计入项目的直接成本，但以 TOC 来评估项目投入才能比较准确地评估出项目的实际代价。

3．效益分析

有了估算出来的开发成本以后，就可以进行效益分析了。在进行效益分析之前，应该首先对系统应用之后，将会带来的直接、间接收益，以及成本降低的具体数额进行量化。并且通过经济学的相关模型来进行分析，这就要求系统分析师能够在经理、管理、数学方面有一定的知识积累。通常进行效益分析时要借助以下几个概念。

（1）货币的时间价值：也就是说现在的 1 元钱，与未来的 1 元钱，其代表的价值是不同的。通常是使用利率（或行业基准率）的形式来表现这个价值。用 F 代表未来的价值，P 代表现在的价值（现值），那么可以总结为：$F=P\times(1+i)^n$ 和 $P=F/(1+i)^n$，其中 i 代表年利率，n 代表年数。

（2）投资回收期：收益的累计数开始超过支出的累计数的时间，也就是说，要多少年才能够将投资回收。显然，投资回收期越短，项目风险就越小。

（3）净现值：整个生命周期内系统的累计经济效益（折成现值）与投资（折成现值）之差。

（4）投资回收率：计算公式是：$P=F_1/(1+j)+F_2/(1+j)^2+\ldots+F_n/(1+j)^n$，其中 P 代表总投资额（折成现值），F_i 是第 i 年年底的净现金（收入减去支出），n 是系统使用年限，j 就是投资回报率。要通过解这个方程来求出 j，是比较困难的一件事情。因此，一般简单地说，投资回收率就等于投资回收期的倒数。

（5）收益/投资比：项目实施后整个系统生命期的收益/投资比值。

（6）敏感性分析：分析项目中的一些关键性因素，例如，系统生命期长度、系统的工作负荷量、工作负荷的类型、处理速度、设备和软件的配置等因素发生变化或进行合理搭配时，对开支和收益的影响最灵敏的范围估算。通常，当项目需要在不同因素之间取舍和调整的时候，需要参考敏感性分析的内容。

有关货币的时间价值、净现值、投资回收期、投资回收率等方面的计算是系统分析师必须要掌握的技术，请阅读本套丛书中的《系统分析师考试全程指导（2009 版）》的第 9.5.2 节。

4．项目可能涉及的收益

项目的收益，通常可能包括一次性经济收益、非一次性经济收益和非经济收益（社会效益）。

（1）一次性经济收益。可能包括开支的缩减、价值的提升。开支的缩减是指改进了的系统的运行所引起的开支缩减。例如，资源要求的减少，运行效率的改进，数据进入、存储和恢复技术的改进，系统性能的可监控，软件的转换和优化，数据压缩技术的采用，处理的集中化/分布化等；价值的提升是指由于一个应用系统的使用价值的提升所引起的收益。例如，资源利用的改进，管理和运行效率的改进以及出错率的减少等。另外，项目的一次性收益可能还包括其他方面的收入，例如从多余设备出售回收的收入等。

（2）非一次性经济收益。在整个系统生命期内，由于运行系统而导致的按月的、按年的能用货币数目表示的收益，包括开支的减少和避免。

（3）非经济收益。非经济收益是无法直接用货币表示的收益，例如，服务的改进、由操作失误引起的风险的减少、信息掌握情况的改进、组织机构给外界形象的改善等。部分非经济收益可以用定性估算的方法或极值分析（最大、最小、乐观、悲观）方式归结到经济效益上。另外，一些非经济收益即使进行估算也非常困难，但常常涉及企业的长期利益。例如，技术积累、对公司业务和产品线的完善和支持、开辟新市场和利润增长点、进入预期能带来较高收益的新市场、提高客户满意度和忠诚度、打击竞争对手抢夺市场份额、获得新的信息化能力从而改善经营或管理格局等。

5. 总结

成本效益分析中的成本分析，将尽可能地列举所有项目涉及的直接财务支出数字，以便管理层协调和制定各种资源的支出计划。成本效益分析中的收益分析，将尽可能清晰地列举实施项目带来的各种直接经济收益和无形收益，以便管理层理解项目的价值和给予项目资源上的支持。否则，一旦项目所需要的各种资源不能按计划投入（资金、人力资源、设备等），项目失败的风险将大大增加，并且除了变更项目预设的开发目标外，几乎没有可供选择的应急方案。

希赛教育专家提示：系统分析师在做成本效益分析的时候，首先是估算新系统的开发成本，然后与可能取得的效益（有形的和无形的）进行比较权衡。有形的效益可以用货币的时间价值、投资回收期、投资回收率等指标进行度量。无形的效益主要是从性质上、心理上进行衡量，很难直接进行量上的比较。

1.4 现有系统的分析

准确地说，对现有系统的分析也是一种需求获取技术（请参考第 2.2 节），只不过这个技术的面比较广，而且对其分析的结果将作为可行性研究过程中的一个重要参照标准。毕竟不管原来的系统还在运行还是已经停用，它与新系统之间总存在着“藕断丝连”的紧密联系，对其进行分析，并与新系统进行比较，就可以获得许多重要的信息。

对现有系统进行分析的一项很重要的工作就是仔细阅读相关的文档资料、使用手册，以及认真地实地考察现有系统。着重了解现有系统完成了什么功能、为什么这样做，

以及现有系统的运行成本、优缺点等。在研究现有系统时千万不要闭门造车，应该多与客户进行沟通，了解他们对现有系统的认识与评价。而且，最重要的是获得他们对现有系统的负面评价，这些问题都将是新系统必须克服和解决的，这些信息对于系统分析师来说，是十分珍贵的。

结合现有系统分析，进行新系统设计的过程如图 1-8 所示。

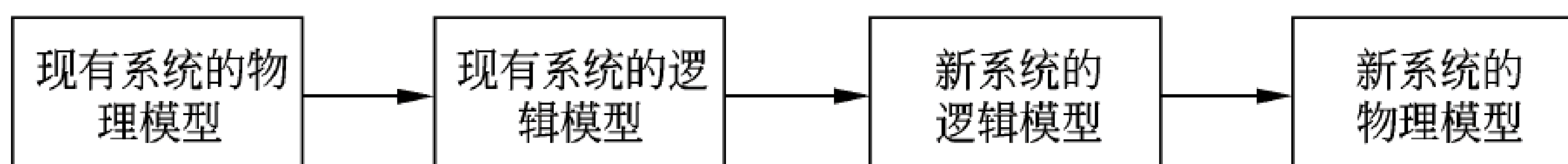


图 1-8　现有系统的研究和分析过程

也就是说，应该从现有系统的物理模型出发，通过研究、分析建立起其较高层的逻辑模型描述。然后，在此基础上吸取各种问题的考虑，发展成为新系统的逻辑模型，再根据新系统的逻辑模型构建出相应的物理模型。

（1）获得现有系统的物理模型。现有系统可能是需要改进的某个已在计算机运行的系统，也可能是一个人工的数据处理过程。在这一步，首先分析、理解现有系统是如何运行的，了解现有系统的组织结构、输入输出、资源利用情况和日常数据处理过程，并用一个具体模型来反映自己对现有系统的理解。这一模型应客观地反映现实世界的实际情况。

（2）抽象出现有系统的逻辑模型。在理解现有系统“怎样做”的基础上，抽取其“做什么”的本质，从而从现有系统的物理模型抽象出当前系统的逻辑模型。在物理模型中有许多物理因素，随着分析工作的深入，有些非本质的物理因素就成为不必要的负担，因此需要对物理模型进行分析，区分出本质的和非本质的因素，去掉那些非本质的因素即可获得反映系统本质的逻辑模型。

（3）建立新系统的逻辑模型。分析新系统与现有系统逻辑上的差别，明确新系统到底要“做什么”，从现有系统的逻辑模型导出新系统的逻辑模型。

（4）建立新系统的物理模型。根据新系统的逻辑模型构建出相应的物理模型。

在对现有系统进行分析的过程中，最容易进入的误区就是在这个地方花费过多的时间，应该把精力放在对整个系统的宏观了解上，而不要过多地去了解系统的细节。否则，就会走入一个陷阱。

1.5　遗留系统的分析

Brodie 和 Stonebraker 对遗留系统（legacy system）的定义如下：遗留系统是指任何基本上不能进行修改和演化以满足新的变化了的业务需求的信息系统。笔者认为，遗留系统应该具有以下特点：

（1）系统虽然完成企业中许多重要的业务管理工作，但已经不能完全满足要求。一般实现业务处理电子化及部分企业管理功能，很少涉及经营决策。

（2）系统在性能上已经落后，采用的技术已经过时。例如，多采用主机/终端形式或小型机系统，软件使用汇编语言或第三代程序设计语言的早期版本开发，使用文件系统而不是数据库。

（3）通常是大型的软件系统，已经融入企业的业务运行和决策管理机制之中，维护工作十分困难。

（4）系统没有使用现代软件工程方法进行管理和开发，现在基本上已经没有文档，很难理解。

在企业信息系统升级改造过程中，如何处理和利用遗留系统，成为新系统建设的重要组成部分。处理恰当与否，直接关系到新系统的成败和开发效率。遗留系统的演化方式可以有很多种，根据系统的技术条件、商业价值以及维护和运行系统的组织特征不同，可以采取继续维护、某种形式的重构或替代策略，或者联合使用几种策略。究竟采用哪些策略来处理遗留系统，需要根据对遗留系统的所有系统特性的评价来确定。

1.5.1 评价方法

对遗留系统评价的目的是为了获得对遗留系统的更好的理解，这是遗留系统演化的基础，是任何遗留系统演化项目的起点。主要评价方法包括度量系统技术水准、商业价值和与之关联的组织特征，其结果作为选择处理策略的基础。

评价方法由一系列活动组成，如图 1-9 所示。

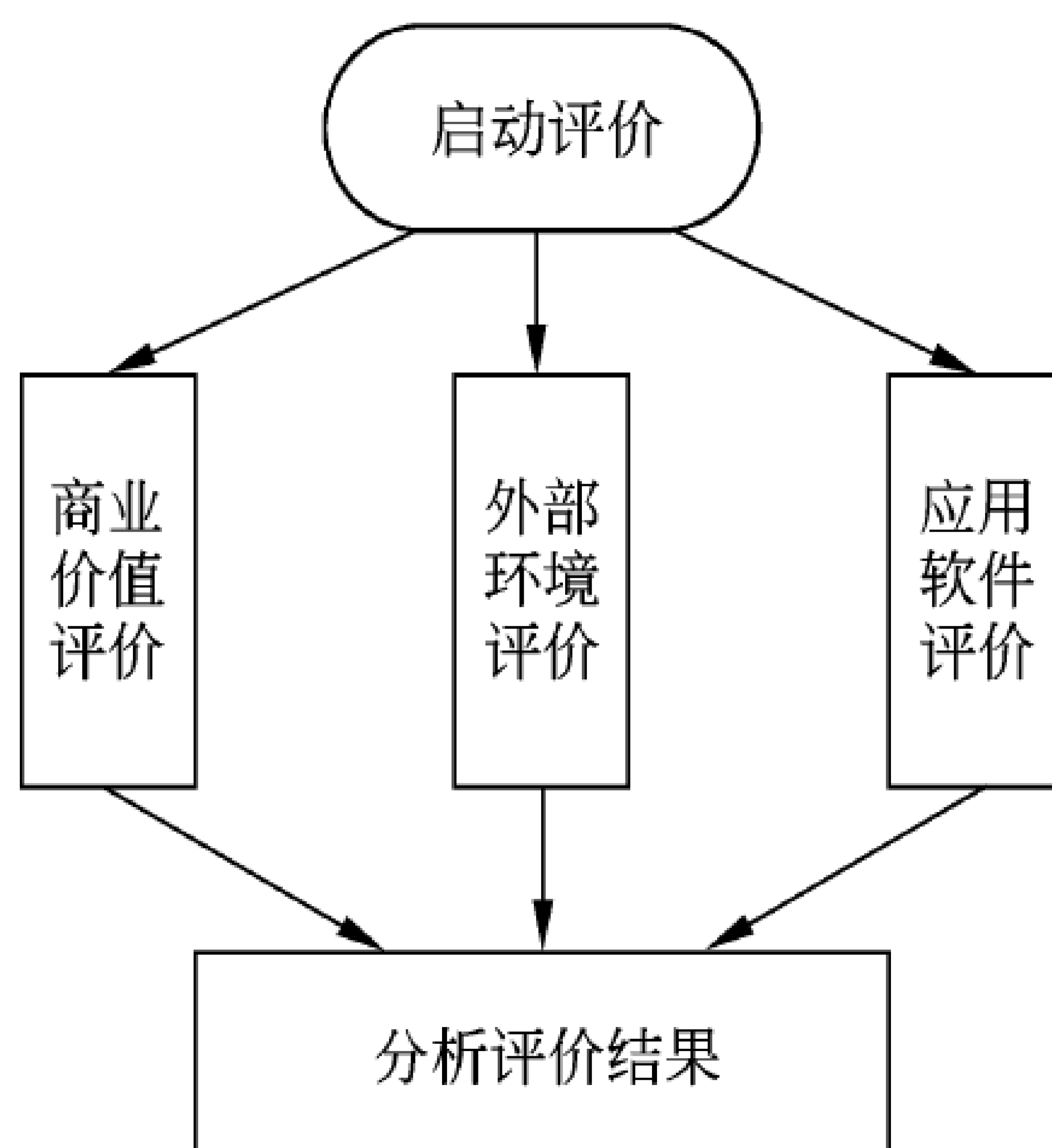


图 1-9 评价活动

1．启动评价

评价是为了获得对遗留系统的足够深度的理解，从技术、业务和企业角度对系统的

理解为系统处理策略提供基础，开始评价前，需要了解以下问题。

（1）对企业来说，遗留系统是否是至关重要的。在评价过程中，可能会发现系统对企业的继续运作产生的影响不大。在这种情况下，就没有必要考虑系统的演化问题。

（2）企业的商业目标是什么。从商业观点来看，系统分析师必须理解企业的商业目标，因为商业目标产生演化需求。

（3）演化需求是什么。演化需求来自企业的商业目标和评价活动。需求必须是可见的，以便决定已存在的系统是否能满足需求。

（4）所期望的系统寿命是多长。一个系统的寿命由软件和硬件的服务能力决定，一旦系统硬件或支撑软件过时，系统的有效性就受到限制。

（5）系统使用期限是多久。如果系统的使用期限只是短期的，就没有必要花费成本来演化系统。相反，如果系统将在相当长的时期内支持主要业务流程，则必须进行演化。

（6）系统的技术状态如何。例如，如果应用软件的技术状况很差，则很难理解，维护成本会很高。

（7）企业是否愿意改变。企业对改变的态度是遗留系统演化成功的关键因素之一。

（8）企业是否有能力承受演化。企业的技术成熟度、员工的素质、支撑工具的级别等都是影响演化的因素。

2．商业价值评价

商业价值评价的目标是判断遗留系统对企业的重要性。在多数情况下，重要业务过程的改变意味着旧的系统现在仅仅具有外围价值，修改这种系统只需花费小许财力和物力。在其他情况下，系统的业务价值很大，需要继续维护运行。可以在概要和详细两个级别上进行遗留系统的商业价值评价。

概要级评价将为更加详细的分析提供信息，包括：

（1）咨询。向有关专家进行咨询，包括最终用户和负责业务处理的管理人员。

（2）评价问卷。问卷应该标识系统在业务处理过程中的哪些地方使用，本系统与其他系统的关系，如果系统不再运行所需的代价，已有系统的缺点和存在的问题等。问题的准确性依赖于所评价的系统。

（3）进行评价。有了问卷的基础后，必须认真分析系统是如何使用的，这往往会发现系统的价值，而这在问卷中是得不到的。

详细级评价包括应用系统不符合业务规范的风险分析，这种分析十分费时，最好由业务分析师来完成详细级的评价工作。

3．外部环境评价

系统的外部技术环境是指硬件、支撑软件和企业基础设施的统一体。

（1）硬件。系统硬件包括许多需要进行常规性维护的部件，这些硬件或者在一个站点，或者分布在许多站点并由网络连接。一般来说，遗留系统的硬件包括主机和小型机、磁盘驱动器、磁带、终端、打印机和网络硬件等。与商业价值评价类似，硬件评价也可

以采用概要级和详细级。概要级评价把遗留系统作为一个整体，提供硬件质量估算。详细评价包括识别系统中的每个部件。在这两种情况下，必须识别一系列特征，用作评价的基础。特征的选择取决于要评价的系统，系统的一些常见特征有供应商、维护成本、失效率、年龄、功能、性能等。

（2）支撑软件。系统的支撑软件环境也由许多部分组成，可包括操作系统、数据库、事务处理程序、编译器、网络软件、应用软件等。一般来说，支撑软件是依赖于某个硬件的，应用软件依赖于系统软件。在评价过程中，必须考虑这种依赖性。

（3）企业基础设施。企业基础设施包括开发和维护系统的企业职责和运行该系统的企业职责（两者可能为同一个企业），这些基础设施是很难评价的，但对遗留系统的演化起关键作用。笔者认为，必须考虑以下问题。

① 企业和使用者的类型。企业或者有自己的软件开发团队，或者所有开发和应用管理都是请其他企业完成。系统用户或许只重复一些记录性工作，或许包括一些更有技术性的工作。

② 开发组织的技术成熟度。开发组织的技术成熟度包括是否使用了现代软件工程方法，是否遵循了统一的标准，是否进行了过程改进等。

③ 企业的培训过程。如果企业（包括开发方和客户方）的培训做得好，遗留系统的演化可能会更成功。

④ 系统支持人员的技术水平。如果系统支持人员的水平和经验不够，就不要急于对系统做大的改动。

⑤ 企业是否愿意改变。企业对改变的态度是遗留系统演化成功的关键因素之一。

支撑软件、企业基础设施的评价方法也类似于硬件评价，在此省略。

4．应用软件评价

应用软件评价可以分为系统级和部件级。

（1）系统级。把整个系统看作是不可分的原子，评价时不考虑系统的任何部分。

（2）部件级。关注系统的每个子系统，考虑每个子系统的特征，包括复杂性、数据、文档、外部依赖性、合法性、维护记录、大小、安全性等。

具体评价方法也与硬件评价类似，在此省略。

5．分析评价结果

评价活动将产生硬件、支撑软件、企业基础设施和应用软件的特征值矩阵，这些特征值体现了遗留系统当前的技术因素，其加权平均值代表了系统的技术水平。计算公式如下：

$$\mathrm{OR} = (P_1\mathrm{ORH} + P_2\mathrm{ORS} + P_3\mathrm{OAF} + P_4\mathrm{ORA})/4$$

其中 ORH 是硬件的评价值，ORS 是支撑软件的评价值，OAF 是企业基础设施的评价值，ORA 是应用软件的评价值，$P_i(1 \le i \le 4)$分别是它们的权系数，即第 i 个评价值对遗留系

统的影响因子。

把对技术水平的全面评价结果与商业评价进行比较，可以为系统演化提供第一手的资料。具体方法是按照商业评价分值和技术水平分值的情况，把评价结果分为 4 种类型，如图 1-10 所示。

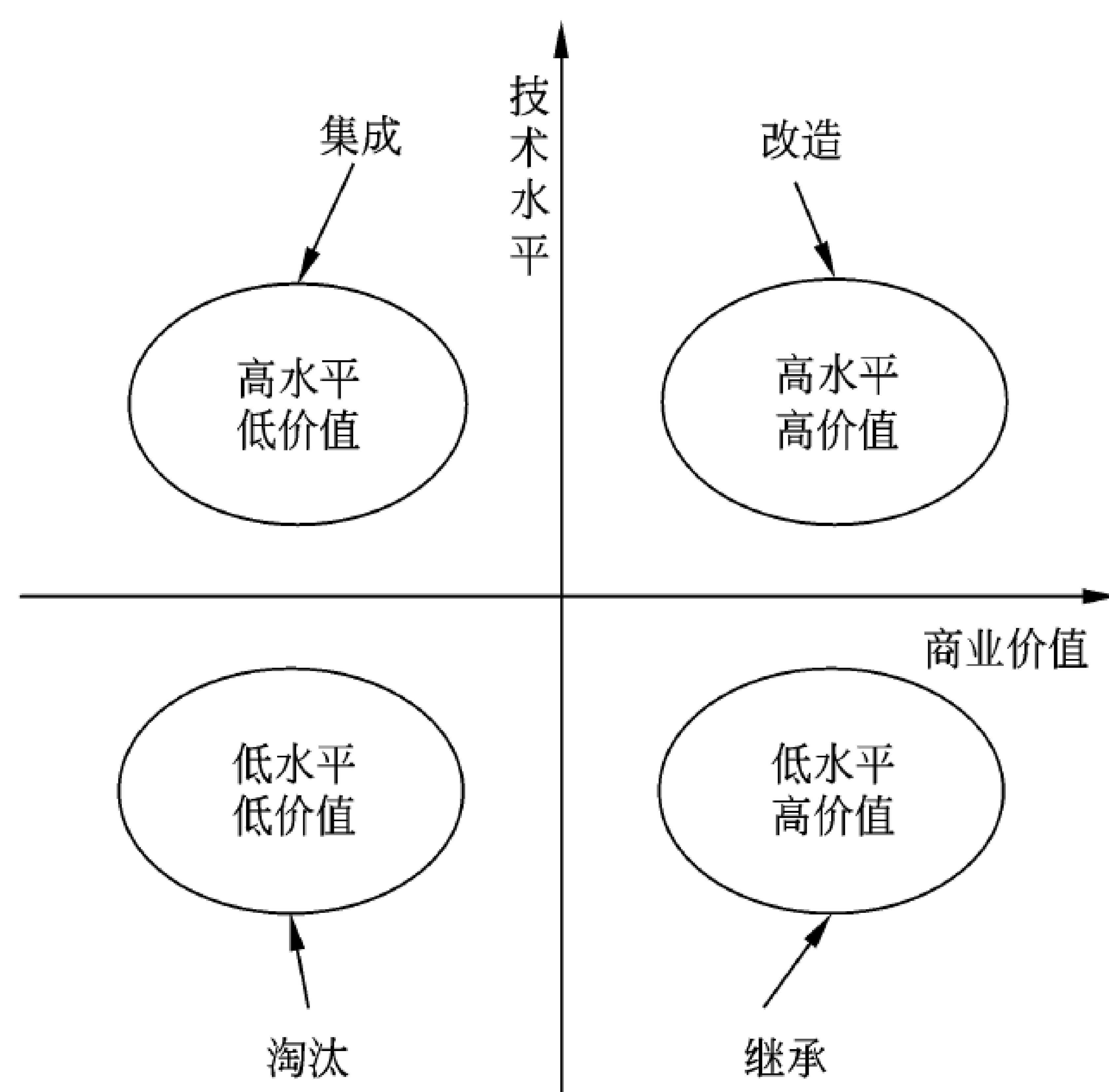


图 1-10　评价结果分析

1.5.2　演化策略

在图 1-10 中，把对遗留系统的评价结果分列在坐标的 4 个象限内。对处在不同象限的遗留系统采取不同的演化策略。

1．淘汰策略

第 3 象限为低水平、低价值区，即遗留系统的技术含量较低，且具有较低的商业价值。对这种遗留系统的演化策略为淘汰，即全面重新开发新的系统以代替遗留系统。

完全淘汰是一种极端性策略，一般是企业的业务产生了根本的变化，遗留系统已经基本上不再适应企业运作的需要；或者是遗留系统的维护人员、维护文档资料都丢失了。经过评价，发现将遗留系统完全淘汰，开发全新的系统比改造旧系统从成本上更合算。

对遗留系统的完全淘汰是企业资源的根本浪费，系统分析师应该善于“变废为宝”，通过对遗留系统功能的理解和借鉴，可以帮助新系统的设计，降低新系统开发的风险。

2．继承策略

第 2 象限为低水平、高价值区，即遗留系统的技术含量较低，已经满足企业运作的功能或性能要求，但具有较高的商业价值，目前企业业务尚紧密依赖该系统。对这种遗

留系统的演化策略为继承。在开发新系统时，需要完全兼容遗留系统的功能模型和数据模型。为了保证业务的连续性，新老系统必须并行运行一段时间，再逐渐切换到新系统上运行。

要做到对遗留系统的继承，必须对系统进行分析，得到旧系统的功能模型和数据模型，这种分析可以部分代替或验证系统的需求分析。如果遗留系统的维护文档已经不完全了，而又必须解析系统的功能模型和数据模型，那将是一项十分艰巨的任务。这时，可使用有关系统重构的 CASE（Computer Aided Software Engineering，计算机辅助软件工程）工具，通过分析系统的代码产生出系统结构图或其他报告。

3．改造策略

第 1 象限为高水平、高价值区，即遗留系统的技术含量较高，本身还有极大的生命力。系统具有较高的商业价值，基本上能够满足企业业务运作和决策支持的要求。这种系统可能建成的时间还很短，对这种遗留系统的演化策略为改造。

改造包括系统功能的增强和数据模型的改造两个方面。系统功能的增强是指在原有系统的基础上增加新的应用要求，对遗留系统本身不做改变；数据模型的改造指将遗留系统的旧的数据模型向新的数据模型的转化。

4．集成策略

第 4 象限为高水平、低价值区，即遗留系统的技术含量较高，但其商业价值较低，可能只完成某个部门（或子公司）的业务管理。这种系统在各自的局部领域里工作良好，但对于整个企业来说，存在多个这样的系统，不同的系统基于不同的平台，不同的数据模型。对这种遗留系统的演化策略为集成。

在集成过程中，可采用由互连系统构成的系统架构，遗留系统可作为从属系统来描述。

希赛教育专家提示：本节所介绍的遗留系统演化策略具有通用性，在实际工程项目中，遇到处理遗留系统的问题时，要具体情况具体分析，选择最佳的演化策略。

1.6 所需要资源估计

对项目所需要的资源进行评估的目的，通常包括如下几个方面：

（1）制订项目所需的资源清单，列举不足的资源，以便提前制定计划和进行准备。

（2）通过合理选择并复用软件构件类资源，降低开发成本、减少交付时间、提高质量。

对资源的描述可以采用统一资源清单的表格，每一类资源至少使用三个特征进行说明。

（1）资源描述：详细说明所需资源的内容和特性。

（2）可用性说明：描述资源当前是否可用、存量，以及当不足的时候可获得的方式。

（3）投入时间：说明开始占用资源的时间，以及需要连续占用的时间。

从信息系统项目所投入的资料来看，大体上可以分为两类，分别是与开发相关的项目资源、与产品/服务相关的资源。

1．与开发相关的项目资源

（1）组织和人力资源。可能包括外部可协作的开发组织、服务和支持性团队、项目开发团队所需要的管理服务性职位（例如，项目经理、产品经理、领域专家、顾问、售前技术支持人员、文档管理员、网络管理员等）和技术性职位（例如，技术经理、高级软件工程师、测试人员、系统分析设计师、美工、内容编辑）等。

对于组织和人力资源还需要重点研究的另一个问题，就是项目开发团队的组织方式。在小型项目中，很多角色可能是重合的（一人身兼数职）；而在较大型的项目中，则需要设置不同的岗位，并严格定义工作职责和工作流程。开发团队的组织方式还可能和企业组织形态、企业文化、项目的目标等有密切关系。有的企业倾向于按项目划分团队的方式进行组织，而另一些企业可能会倾向于建立层级型组织，将所有项目集中开发和管理。

（2）软件资源。主要是指各种不同程度可复用的软件资源。这些资源可能包括可直接使用的第三方构件或开发平台，以及在某些类似项目中建立的规约、设计、代码、测试数据、方案、需求报告等。

在项目开发论证阶段，应将复用现有的软件资源提到一个特别重要的程度来看待。通常复用现有的软件构件的程度越高，项目的成本、开发时间、质量等就越能得到改善。

希赛教育专家提示：*在系统设计中，也应该特别关注那些未来可能复用的知识、方案、代码或构件等，在当前项目中对这些资源进行前瞻性的设计与开发，将在未来很多的类似项目中得到回报。*

（3）开发环境。主要是指支持开发团队的各种环境、设备和工具。例如，测试环境、试运行环境的各种软件和硬件设备；用于项目管理、知识管理、测试、缺陷跟踪软件、变更跟踪软件、需求管理软件、建模工具、开发语言和开发平台等各种支持性或工具性软件。

2．与产品和服务相关的资源

（1）产品售后技术支持人员和客户支持人员。

（2）产品化相关的资源。例如，宣传、支持、服务、培训用的各种文档，软件包装、标签、加工等所需要的管理人员、物品、美工设计等。

（3）外部采购或合作涉及的商务人员、合同、文书模板；法律顾问或相关人员；软件版权申请和登记所需要的各种资源和准备性工作等。

（4）根据项目内容提供给销售人员的样品、宣传页、手册、演示盘、宣传性网站等。

1.7 现有资源的有效利用

充分利用现有的软件、硬件（以下统称为“软硬件”）和数据资源，可以尽量保护现有 IT 设施的投资，减少升级系统时的直接成本（例如，购买新的设备）、间接成本（例如，培训、安装软件系统等）和对业务的影响，在新系统中导入和利用具有重要价值的业务数据。

为了达到充分利用现有的软硬件和数据资源的目的，可在多方面采取措施：

（1）在设计中降低对硬件和基础平台的要求。各种技术手段均可以降低对软硬件的要求，从而使得原有的软硬件可以继续满足新系统的配置要求。例如，在数据处理中将每笔交易分散、分布处理；不采用集中式计算统计；使用更高效的算法、优化数据库设计和数据减少冗余，采用可支持用户分阶段进行 IT 设施投入的技术（例如，负载均衡技术），不但能够降低当期的软硬件采购成本，还能够让未来的采购更有价值，以及保留更多的选择。

（2）调整体系结构和设计手段。例如，从 C/S（Client/Server，客户/服务器）架构向 B/S（Browser/Server，浏览器/服务器）架构迁移，或多层分布式计算架构，或采用中间件服务器完成主要的计算工作，可大大降低对客户端软硬件的系统需求，从而达到利用现有资源的目的。

（3）支持数据标准。通过支持各种数据存储、描述、交换、处理的标准，可以保证新旧系统转换或协作时的扩展性，数据资源能够用通行的格式被处理，而不需要在升级系统时进行迁移和转换。

（4）采用兼容的软件、硬件和体系结构。在 IT 建设中，尽量采用兼容的软件、硬件和相同的体系结构，不仅可以大大降低成本，保证系统的扩展性，而且使未来利用这些软硬件的可能性大大增加。

充分利用软硬件和数据资源，只是人们试图控制 IT 成本的努力之一。由于计算机软硬件技术进步的速度非常快，旧有的软硬件如果不能在采购后充分利用，很快就将面临贬值。因此，降低软硬件成本，充分利用现有设施的重要性在逐步降低，而保护业务数据、增值业务数据的需求和重要性，则远远高于充分利用原有软硬件的需求和重要性。在很多情况下，仅考虑如何利用现有的软硬件节省成本，常常只能得到非常有限的效果，或者根本抓不住问题的要点所在。

在 20 世纪 70～80 年代，信息系统生命周期预期为 10 年或 20 年，但在今天高度竞争、易变化的环境中，类似电子商务这样的软件系统，从软硬件投资、系统开发建设、应用运营的总生命周期或许只能持续 1～2 年，然后就被废弃或完全重写。在这种情况下，考虑原有软硬件利用或目前建设的 IT 设施未来能被尽量利用的目标，将变得毫无意义。现有软硬件和数据资源有效利用的意义，在于尽量“物尽其用”。从系统设计方法的角度

看，对于这类系统，在系统建设前即采用面向成本的设计方法，例如，DTC（Design To Cost，在有限度的资源内设计）和 DFC（Design For Cost，尽可能减少成本的设计），或者采用满意质量策略（Good Enough Quality，GEQ）的设计方法，进行事前估算而不考虑“过时”后的利用，或许是更加有效的策略。

另一方面，在电信、保险、金融、电力等曾长期进行过 IT 基础建设的领域中，过去巨额投资的 IT 设施依然在发挥极其重要的作用。尽管其中的软硬件、设计方法、开发工具等在今天看来早已过时，但推翻重来的影响和代价常常是不可接受的，新的系统必须尽量和原有系统保持一致。对这些系统来说，现有软硬件和数据资源的有效利用的意义则在于，保证系统稳定、持久地运行和维护，而不在于重建系统后，如何利用原有设施。

希赛教育专家提示：充分利用现有资源的努力，必须放在系统整个生命周期的过程中进行考察，才能发现其价值。而持续的成本控制和优化的目标，也必须作为一个系统工程，在通盘考虑运营、规划、维护、系统设计等多方面的努力下，才能得到满意的解决。

1.8　系统方案的制定

通过在问题定义和归结模型阶段的工作，已经分析并定义了与系统开发目标相关的各种模型，分析出了系统的功能清单、性能要求等，解释了“系统目标是什么”的问题。在系统方案阶段，主要完成的工作则是解释“系统如何实现”的问题。具体来说，系统方案制定阶段要完成的工作包括确定系统架构、实现系统所需要的各种关键性设计要素和实现手段，归结系统目标到最适合的系统架构，将系统功能和关键要素分配到系统架构上。

1．确定系统架构

在问题定义阶段得到的系统概念模型，使用各种工具定义了项目的开发目标。在系统方案制定阶段才开始真正考虑如何去实现系统。其中最重要的工作就是制定系统的实现架构。

（1）软件架构。通过使用软件架构技术的各种逻辑视图、进程视图、物理视图、开发视图和场景视图，具体描述软件系统的高阶抽象模型，揭示系统需求和构成系统的元素之间的对应关系，指导后续软件的分析。有关软件架构的详细知识，请阅读本套丛书中的《系统分析师技术指南（2009 版）》的第 11 章。

（2）关键的设计要素。若干对应于目标的最基本、最重要的实现要素的分析和设计，这些实现要素对应于系统目标最重要的场景，表示整个系统最主要的控制流程和实现机制。

关键性的设计要素可能包括关键的用例、最主要的控制类、功能的组织和调度方式；系统的分层方式、接口、协议、标准等；对象或程序模块（结构化方法）的组织模式；

常用和最关键的算法模型，重要的设计、分析、开发标准和规范。

关键性的实现手段可能包括选定基础计算平台（例如，操作系统、数据库、Web 服务器、中间件平台等），选定开发工具和开发环境（例如，计算机语言、构件库、工具软件等），确定项目的组织方法、管理要件（例如，配置管理、需求管理、知识管理）方案，确定软件过程方法（例如，统一过程、敏捷方法）。

（3）特性和要点的解释。这些附加的内容解释系统的一些特性、服务等是如何实现的。

2．归结系统目标到最适合的系统架构

通常，总是有一些标准的架构可供选择，对于大多数开发项目来说，比较各种标准的架构与预期目标之间的匹配程度即可选定系统架构。选择标准的架构去实现系统，可以忽略大多数基础平台和底层支撑技术的实现问题，从而大大提高系统的质量、降低开发风险和成本。可以根据基础平台的功能和性能指标、企业或项目组的技术积累、待开发系统的特点和分层方式等选定标准的系统架构。

在另一些情况下，出于各种诸如用户指定、与用户现有的 IT 设施保持一致性、兼容性、扩展性、未来维护的能力等因素，系统的基础平台很可能在项目的论证阶段就已经被确定。在这种情况下，系统的开发平台和架构实际上已经确定。

3．将系统功能和关键要素分配到系统架构上

通过将系统功能清单和实现的各种关键要素整理并分类，然后与现有的技术、标准的实现体系进行比较和匹配，就可以将系统概念模型定义的开发目标，进一步映射到真正可计算、可实现的系统架构上。

这个过程可以理解为一种不断归结、比较并匹配、设计以及分配的过程。进行匹配的过程常常是一种双向的选择和探究过程。一旦完成了比较、匹配、分配、确认的过程，也确认了从总体设计到具体模块开发的各层次上需要完成的工作内容。

（1）.NET 实现的案例。.NET 的标准三层架构包括表示层、业务逻辑层和数据服务层。表示层是用户的界面部分；业务逻辑层负责处理表示层的应用请求，完成业务逻辑的计算任务，并将处理结果返回给用户。业务逻辑层是将原先置于客户端的业务逻辑分离出来，集中置于服务器部分，为所有用户共享，是整个应用的核心部分；数据服务层为应用提供数据来源。和以往的两层架构不同，数据库不再和每个活动客户保持一个连接，而是若干个客户通过应用逻辑构件共享数据库的连接，从而减少了连接次数，提高了数据库服务器的性能和安全性。

在这种三层系统结构下，开发不同的系统也会有不同的抉择。例如，当利用数据库存储过程技术来实现业务逻辑的时候，存储过程被看作业务逻辑层的一部分，这是追求系统性能的大型数据库应用系统开发，或绑定在某种数据库平台开发的典型做法。但如果利用 ActiveX 数据对象（ActiveX Data Object，ADO）像访问数据时，ADO 和后台的数据库系统以及数据库的逻辑则被看作数据服务层的一部分。这是系统性能要求不高，

追求软件可移植性、桌面软件系统的典型做法，即仅仅将数据库系统视为数据存储而不采用任何特性。

在有必要的情况下，某个层次还可能进一步细分，例如，把业务逻辑层划分为基本的计算对象层、业务对象层，以及粘合业务对象的“胶水”脚本（Script）或控制调度对象层。

（2）采用 Java 服务器页面（Java Server Page，JSP）技术的 B/S 系统案例。这种系统架构也可以归结为标准的三层架构。其中 JSP 的超文本标记语言（HyperText Mark-up Language，HTML）程序页面构成了前端的表示层，JavaBeans 构成了业务逻辑层，JDBC（Java DataBase Connectivity，Java 数据库连接）和后台的数据库则构成了数据服务层。

对于小规模的网站系统，开发者可能直接在 JSP 页面中书写所有的应用逻辑脚本（Java 程序、HTML 页面、Javascript 脚本，以及 CSS 层次样式表），这样，业务逻辑层就和表示层合并了；开发者的另一种选择，是在 JavaBeans 中封装全部的业务逻辑，然后在 Beans 中提供标准的调用接口。这时，JSP 页面扮演了与用户交互的表示层角色，以及粘合业务逻辑层和表示层的胶水；JavaBeans 构成了业务逻辑层，后端的 JDBC 和数据库系统构成了数据服务层。

对于使用 Java2 平台企业版（Java 2 Platform Enterprise Edition，J2EE）体系的软件人员来说，利用企业 Java Bean（Enterprise Java Bean，EJB）的容器、对象操作语言等机制直接实现了对象级的接口，开发人员可以直接在业务逻辑层去构思应用，JDBC 和后台数据库系统的数据服务层被隐含在 J2EE 的平台机制内，在更高的抽象级别上被屏蔽。

有关.NET 和 J2EE 平台的详细比较，请阅读本套丛书中的《系统分析师技术指南（2009 版）》的第 2 章。

可以看到，即使是同一种抽象的逻辑设计模型，在具体映射到真实系统架构的时候，也需要根据具体的情况进行调整。因此，归结实现要素到系统架构的时候，要点在于理解各种系统架构的大致分层和构成，比较实现要素的目标和实现手段之间的“适合程度”，强调运用而非生搬硬套某种实现机制，甚至不顾系统高阶的需求和约束，盲目追求某种“流行的、先进的”系统架构。

希赛教育专家提示：一个全面的设计是在头脑中不断演算、细化、规划、评估、分配的快速过程，也是高度实现技术导向的。为了完成这一过程，系统分析师必须对流行的系统架构、实现技术、工具/方法和模式、实现难度和代价等了如指掌，还必须具有一种“既见树木又见森林”的兼顾总体把握和细微探究的能力，才能完成系统方案的制定、评价和改进。

本章参考文献

[1]　小约翰・珀西科，帕特里克亚・让娜・莫里丝. 5I 商业价值观——21 世纪经理

人成功的基石. 北京：机械工业出版社，2001

[2] Michael Porter 竞争优势. 陈小悦译. 北京：华夏出版社，2003

[3] 张友生. 遗留系统的评价方法和演化策略. 计算机工程与应用，2003（13）：29-35

[4] 张友生. 软件体系结构. 第 2 版. 北京：清华大学出版社，2006

[5] 郑人杰. 实用软件工程. 北京：清华大学出版社，1997

第 2 章　需求获取与分析

Standish Group 曾经对 23 000 个项目进行研究，其结果表明，28%的项目彻底失败，46%的项目超出经费预算或者超出工期，只有约 26%的项目获得成功。需求开发在整个软件生命周期中有着十分重要的意义，在高达 74%的不成功项目中，有约 60%的失败是源于需求问题。也就是说，有将近 50%的项目都遇到了需求的问题。

需求开发是软件开发过程中最有关键的一个环节，可以说项目成败的关键就在于此。然而，对于大部分客户而言，由于对计算机知识的缺乏，通常都无法明确阐述需求，这就要求系统分析师能够熟练掌握这方面的技能。

2.1　需求的分类

简单地说，软件需求就是系统必须完成的事以及必须具备的品质。具体来说，软件需求包括功能需求、非功能需求和设计约束等三方面的内容。

（1）功能需求：是指系统必须完成的那些事，即为了向它的用户提供有用的功能，产品必须执行的动作。

（2）非功能需求：是指产品必须具备的属性或品质，如可靠性、性能、响应时间、容错性、扩展性等等。

（3）设计约束：也称为限制条件、补充规约，通常是对解决方案的一些约束说明，例如，必须采用国有自主知识产权的数据库系统，必须运行在 UNIX 操作系统之下等。

另外，在大量与需求相关的书籍、文章中有一些诸如业务需求、用户需求之类的词语，把很多读者搞得术语混淆，下面，一起来看看这些概念。

（1）业务需求（Business Requirement）：是指反映组织机构或客户对系统、产品高层次的目标要求，通常问题定义本身就是业务需求。

（2）用户需求（User Requirement）：是指描述用户使用产品必须要完成什么任务，怎么完成的需求，通常是在问题定义的基础上进行用户访谈、调查，对用户使用的场景进行整理，从而建立从用户角度的需求。

（3）系统需求（System Requirement）：是从系统的角度来说明软件的需求，包括用特性说明的功能需求、质量属性，以及其他一些非功能需求、设计约束等。

也就是说，这分别对应于需求的三个不同层次，从目标到具体，从整体到局部，从概念到细节。这些不同层次、不同类型的需求描述之间的关系如图 2-1 所示。

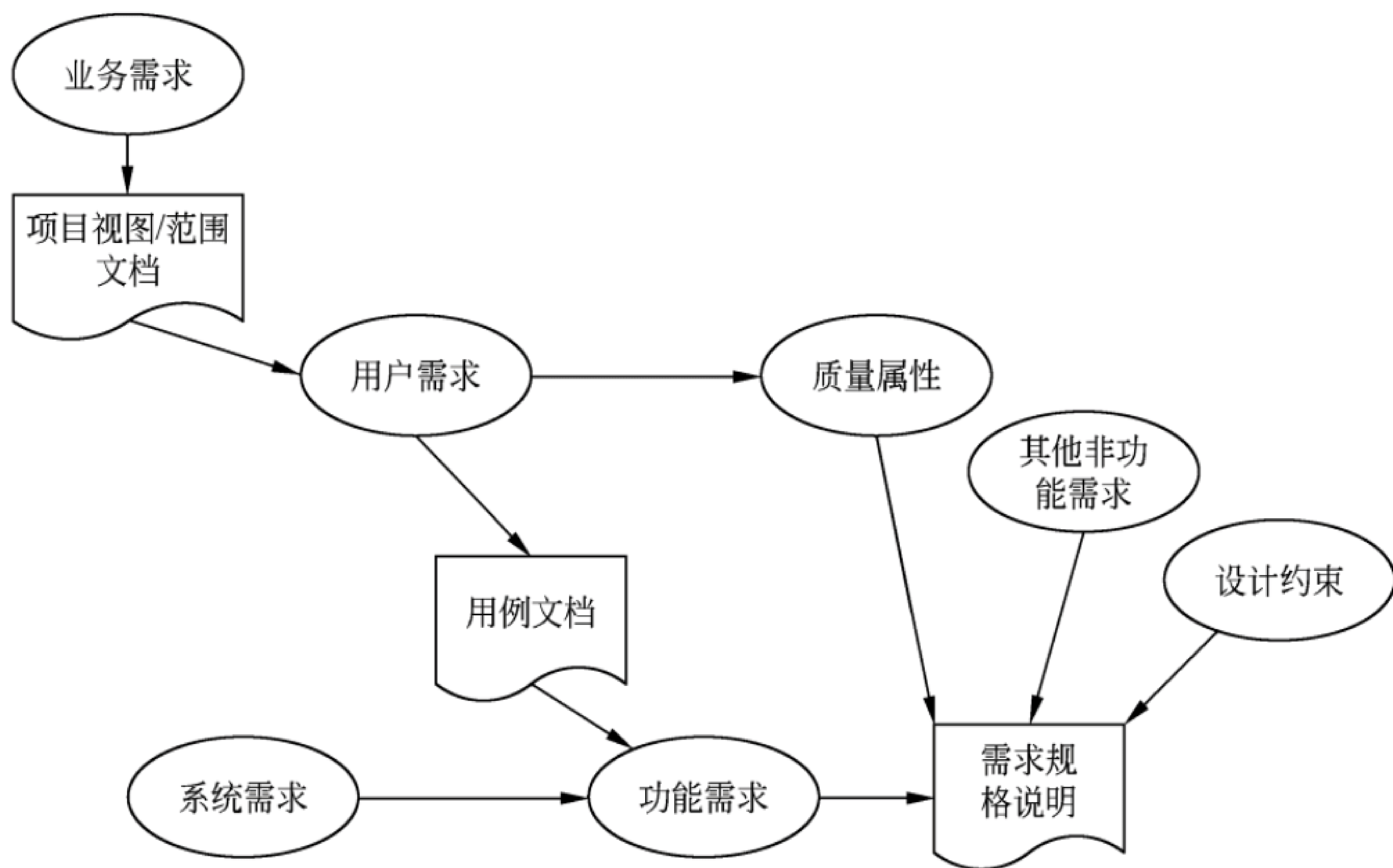


图 2-1 需求的组成结构示意图

2.2 需求获取的方法

需求调查与问题定义是一件看上去很简单，做起来却很难的事情。需求调查是否科学、准备充分，对调查出来的结果影响很大，这是因为大部分客户无法完整地讲述需求，而且也不可能看到系统的全貌。要想做好需求调查，必须清楚地了解以下三个问题。

（1）What：应该搜集什么信息。

（2）Where：从什么地方搜集这些信息。

（3）How：用什么机制或者技术来搜集这些信息。

接下来，就对这三个部分的内容做一些更加详细的说明与描述。

1. 要获取的信息

一方面，系统分析师应该知道，从宏观的角度来看，要获取的信息包括三大类：一是与问题域相关的信息（例如，业务资料、组织结构图、业务处理流程等），二是与要求解决的问题相关的信息，三是用户对系统的特别期望与施加的任何约束信息。这样才能够有的放矢，不会顾此失彼。

另外一方面，系统分析师在开展具体需求获取工作时，应该做到在此之前明确自己需要获得什么信息，这样才有可能获得，才知道是否顺利完成目标。

2. 信息的来源

除了要明确地知道需要什么方面的信息，还要知道它们可以从哪里获得。而通常情

况下，这些需要的信息会藏于客户、现有系统、现有系统用户、新系统的潜在用户、现有产品、竞争对手的产品、领域专家、技术法规与标准里。

面对这么许多种可能，在具体的实践中该从何下手呢？其实也很简单。首先，从人的角度来说，应该首先对项目干系人（风险承担者、涉众）进行分类。然后，从每一类项目干系人中找出 1 或 2 名代表；而对于文档、产品而言，则更容易有选择地查阅。

希赛教育专家提示：在制定需求获取计划的时候，不妨列出一个表格，左边写上想了解的信息，右边写上信息可能的来源。这样，就能够建立起一一对应的关系，使需求获取工作更加高效。

3. 需求获取技术

作为一名系统分析师，掌握各种不同的需求获取技术，并且熟练地在实践中运用它，是十分必要的。在此，列举出一些最常用的需求获取技术。

1）用户访谈

用户访谈是最基本的一种需求获取手段，其形式包括结构化和非结构化两种。结构化是指事先准备好一系列问题，有针对性地进行；而非结构化则是只列出一个粗略的想法，根据访谈的具体情况发挥。最有效的访谈是结合这两种方法进行，毕竟不可能把什么都一一计划清楚，应该保持良好的灵活性。

（1）准备问题：进行用户访谈之前，最好先对要询问的问题进行一些准备。准备的方法是围绕想要获取的信息展开，设计一系列的问题，按顺序组织起来。而且，还要预先准备好记录方式，主要包括本人记录、第三人记录或者是录音/录像的形式。如果采用录音/录像的方式，应该征得被访谈者的同意。这种方法虽然看上去比较有效，不容易丢失信息，但这也会给后面的整理工作带来一定的工作量和难度。

（2）访谈时的技巧：在访谈时一定要注意措词得当，在充分尊重被访谈者的基础上，要避免出现“我不知你在说什么”、“我是来帮助你更好的工作”这样的言语。否则，将会破坏访谈的气氛，从而使访谈的效率大打折扣。在访谈时，一定要注意保持轻松的气氛，选择客户有充裕空闲的时段进行，在说话、提问时应该尽量采用易于理解、通俗化的语言。另外，值得注意的是，系统分析师应该在进行访谈之前进行一些领域相关的知识培训，充分阅读相关材料，以保证自己有较专业的理解与认识，让被访谈者能够信任自己。

（3）应该询问的问题：在设计询问的问题时，应该问自己：我的问题看起来相关吗？回答是否正式？对方是回答这些问题的合适人选吗？是否问了过多的问题？是否还有更多的问题要问被访者？另外，还可以在过程中询问被访者还希望自己问他什么问题，还应该见谁。

总的来说，用户访谈具有良好的灵活性，有较宽广的应用范围。但是，也存在着许多困难，例如，客户经常较忙，难以安排时间；面谈时信息量大，记录较为困难；沟通需要很多技巧，同时需要系统分析员有足够的领域知识。另外，在访谈时，会遇到一些

对于组织来说比较机密和敏感的话题。因此，这看似简单的技术，也需要系统分析人员拥有足够多的经验和较强的沟通能力。

2）用户调查

用户访谈时最大的难处在于很多关键的人员时间有限，不容易安排过多的时间。而且，客户面经常较广，不可能一一访谈。因此，就需要借助“用户调查”这一方法，通过精心设计要问的问题，然后下发到相关的人员手中，让他们填写答案。这样，就可以有效地克服用户访谈方法中存在的问题。

与用户访谈相比，用户调查最大的不足就是缺乏灵活性；双方未见面，系统分析人员无法从用户的表情等其他动作来获取一些更隐性的信息；客户有可能在心理上会不重视一张小小的表格，不认真对待从而使得反馈的信息不全面。

因此，较好的做法是将用户访谈和用户调查结合使用。具体来说，就是先设计问题，制作成为用户调查表，下发填写完后，进行仔细的分组、整理和分析，以获得基础信息。然后，再针对分析的结果进行小范围的用户访谈，作为补充。

3）现场观摩

俗话说得好：“百闻不如一见”。对于较为复杂的流程和操作而言，是难以用言语表达清楚的，即使用言语表达，也会显得很低效。因此，针对这一现象，系统分析师可以就一些较复杂、较难理解的流程和操作，采用现场观摩的方法来获取需求。具体来说，就是走到客户的工作现场，一边观察，一边听客户的讲解，甚至可以安排人员跟随客户工作一小段时间。这样，就可以更加直观地理解需求。

4）文档考古

对于一些数据流比较复杂的，工作表单较多的项目，有时是难以通过说或者通过观察来了解需求细节的。这时，就可以借助于“文档考古”的方法，也就是对历史存在的一些文档进行研究，“考古”一词正是形象地说明了其主要的工作重心是结合已经填写完毕的、带有数据的文件、表单、报告，从中获得所需的信息。

不过，当准备采用该方法时，也要记住这个方法的主要风险，那就是历史的文档可能与新系统的流程、数据有一些不吻合的地方，并且还可能有错误。要想有效地避免和发现这些问题，就需要系统分析师能够运用自己的聪明才智，将其与其他需求获取技术结合对照。还有一个负面因素就是，这些历史文档中记载的信息有可能涉及到客户的商业秘密，因此，对客户数据信息的保密也是系统分析师基本的职业道德。

5）联合讨论会

这是一种相对来说成本较高的需求获取方法，但也是十分有效的一种。它通过联合各个关键客户代表、系统分析师、开发团队代表一起，通过有组织的会议来讨论需求。通常该会议的参与人数为 6～18 人，召开时间为 1～5 小时。

在会议之前，应该将与讨论主题相关的材料提前分发给所有将要参加会议的人。在会议开始之后，首先应该花一些时间让所有的与会者互相认识，以使交流在更加轻松的

气氛下进行。会议的最初，就是针对所列举的问题进行逐项专题讨论；然后，对现有系统、类似系统的不足进行开放性交流；第三步则是大家在此基础上对新的解决方案进行一番设想，在这个过程中，需要把这些想法、问题、不足记录下来，形成一个要点清单；第四步就是针对这个要点清单进行整理，明确优先级，并进行评审。

联合讨论会将会起到群策群力的效果，对于一些问题最歧义的时候、对需求最不清晰的领域都是十分有用的一种方法。这种方法最大的难度就是会议的组织，要做到言之有物，气氛开放。否则，将难以达到预想的效果。

4. 需求获取的策略

在整个需求开发的过程中，需求获取、需求分析、需求定义、需求验证 4 个阶段不是瀑布式的发展，而是应该采用迭代式的演化过程。也就是说，在进行需求获取时，不要期望着一次就将需求收集完，而是应该获取到一些信息后，进行相应的需求分析，并针对分析中发现的疑问和不足，带着问题再进行有针对性的需求获取工作。

2.3　需求分析的任务

所谓分析，就是通过对问题域的研究，获得对该领域特性及存在于其中（需要解决）的问题特性的透彻理解并用文档说明。需求分析就是提炼、分析和仔细审查已经获取到的需求，以确保所有的项目干系人都明白其含义并找出其中的错误、遗漏或其他不足的地方。需求分析的关键就在于对问题域的研究与理解。为了便于理解问题域，现代软件工程方法所推荐的做法就是对问题域进行抽象，将其分解为若干的基本元素，然后对元素之间的关系进行建模。

Karl E.Wiegers 在《软件需求》一书中指出，需求分析的工作通常包括以下 7 个方面。

（1）绘制系统上下文范围关系图：这种关系图是用于定义系统与系统外部实体间的界限和接口的简单模型，它可以为需求确定一个范围。

（2）创建用户界面原型：用户界面对于一个系统来说是十分重要的，因此在需求分析阶段通过快速开发工具开发一个可抛弃原型，或者通过 PowerPoint、Authware 等演示工具制作一个演示原型，甚至是用纸笔画出一些关键的界面接口示意图，将帮助客户更好地理解所要解决的问题，更好地理解需求。有关快速开发工具的详细知识，请阅读本套丛书中的《系统分析师技术指南（2009 版）》的第 22 章。

（3）分析需求的可行性：对所有获得的需求进行成本、性能、技术实现方面的可行性研究，以及这些需求项是否与其他的需求项有冲突，是否有什么对外的依赖关系等。

（4）确定需求的优先级：这是一项很重要的工作，迭代开发已经成为了现代软件工程方法论的一个基础，而需求的优先级是制订迭代计划的一个最重要的依据。对于需求优先级的描述，可以采用满意度/非满意度指标进行说明。其中满意度可以取值 1～5，表示当需求被实现时用户的满意程度；不满意度可以取值 1～5，表示当需求未被实现时

用户的不满意程度。

（5）为需求建立模型：也就是建立分析模型，这些模型的表现形式主要是图表加上少量的文字描述，正所谓“一图抵千字”，图形化地描述需求将使得其更加清晰、易懂。根据采用的分析技术的不同，采用的图也将不同。例如，面向对象技术下的用例模型和域模型，面向数据分析方法下的 E-R 图，结构化分析方法下的 DFD 等。更多的内容，请阅读第 2.5 节。

（6）创建数据字典：数据字典是对系统用到的所有数据项和结构进行定义，以确保开发人员使用了统一的数据定义。

（7）使用质量功能展开（Quality Function Deployment，QFD）：这是在需求优先级基础上的一个升华，其原理与满意度/非满意度指标十分接近。通过将产品特性、属性与对客户的重要性联系起来，QFD 将需求分为三类，分别是期望需求（如果缺少，会让用户感到不满意）、普通需求和兴奋需求（如果实现了，客户会感到惊喜；如果没有实现，也不会遭到责备）。

希赛教育专家提示：仅仅单一来看需求并不能提供对需求的完全理解，因此，需要把用文字表示的需求和用图形表示的需求结合起来。而用图形来表示需求，就是需求建模，获得分析模型。分析模型将有助于检测需求的不一致性、模糊性、错误及遗漏。

2.4 需求分析方法论

需求分析的方法种类繁多，不过，如果按照分解的方式不同，可以很容易地划分出几种大类型。

（1）结构化分析方法（Structured Analysis，SA）：最初的分析方法都不成体系，而且通常都只包括一些笼统的告诫，在 20 世纪 70 年代，分析技术发展的分水岭终于出现了。这时，人们开始尝试使用标准化的方法，开发和推出各种名为结构化分析的方法论，而 Tom DeMacro 则是这个领域最有代表性和权威性的专家。

（2）软系统方法：这是一个过渡性的方法论，并未真正流行过，它的出现只是证明了 SA 方法的一些不足。因为 SA 方法采用的相对形式化的模型不仅与社会观格格不入，而且在解决不确定性时显得十分无力。最有代表性的软系统方法是 Checkland 方法。

（3）面向对象分析方法（Object Oriented Analysis，OOA）：在 20 世纪 90 年代，SA 方法的不足在面对多变的商业世界时，显得更加苍白无力，这就催生了 OOA 的迅速发展。

（4）面向问题域的分析（Problem Domain Oriented Analysis，PDOA）：现在，又发现 OOA 方法也存在着很多的不足，应运而生了一些新的方法论，PDOA 就是其中一种。不过，这些方法目前还处在“实验室”阶段，并未得到广泛应用。

2.4.1　结构化分析

SA 方法与 OOA 方法之间的最大差别是：SA 方法把系统看作一个过程的集合体，包括人完成的和计算机完成的；而 OOA 方法则把系统看作一个相互影响的对象集。SA 方法的特点是利用 DFD 来帮助人们理解问题，对问题进行分析。

SA 方法一般包括 DFD、数据字典、结构化语言、判定表和判定树等工具。从总体上来看，SA 是一种强烈依赖 DFD 的自顶向下的建模方法，它不仅是需求分析技术，也是完成需求定义的有效技术手段。

1. 工作步骤

在介绍具体的 SA 方法之前，先对如何进行结构化分析做一个总结性描述，以帮助读者更好地应用该方法。

（1）研究“物质环境”。首先，应画出现有系统的 DFD，说明系统的输入/输出数据流，说明系统的数据流情况，以及经历了哪些处理过程。在这个 DFD 中，可以包括一些非计算机系统中数据流及处理的命名，例如，部门名、岗位名、报表名等。这个过程将可以帮助系统分析师有效地理解业务环境，在与用户的充分沟通与交流中完成。

（2）建立系统逻辑模型。当物理模型建立完成之后，接下来的工作就是画出相对于真实系统的、与物理模型等价的逻辑 DFD。在前一步骤建立的 DFD 的基础上，将所有自然数据流都转成等价的逻辑流。例如，将现实世界的报表改成为存储在计算机系统中的文件里，将现实世界中“送往总经理办公室”改为“报送报表”。

（3）划清人机界限。最后，确定在系统逻辑模型中，哪些将采用自动化完成，哪些仍然保留手工操作。这样，就可以清晰地划清系统的范围。

2. 数据流图

DFD 是一种图形化的系统模型，它在一张图中展示信息系统的主要需求，即输入、输出、处理（过程）、数据存储。由于从 DFD 中可以很容易地一眼看出系统紧密结合的各个部分，而且整个图形模式只有 4 个符号需要记忆，因此，深受系统分析师的喜爱，广为流行。

在 DFD 中，通常会出现 4 种基本符号，分别是数据流、加工、数据存储和外部实体（数据源及数据终点）。数据流是具有名字和流向的数据，在 DFD 中用标有名字的箭头表示。加工也称为处理、过程，是对数据流的变换，一般用圆圈表示。数据存储是可访问的存储信息，一般用两条直线段表示。外部实体是位于被建模的系统之外的信息生产者或消费者，是不能由计算机处理的成分，它们分别表明数据处理过程的数据来源及数据去向，用标有名字的方框表示。

（1）数据流图的层次。正如前面提到的，结构化分析的思路是依赖于数据流图进行自顶而下的分析。这也是因为系统通常比较复杂，很难在一张图上就将所有的数据流和加工描述清楚。因此，数据流图提供一种表现系统高层和低层概念的机制。也就是先绘

制一张较高层次的数据流图，然后在此基础上，对其中的过程（处理）进行分解，分解成为若干独立的、低层次的、详细的数据流图，而且可以这样逐一的分解下去，直至系统被清晰地描述出来。

（2）Context 图。Context 图也就是系统上下文范围关系图，是描述系统最高层结构的 DFD。它的特点是将整个待开发的系统表示为一个加工，将所有的外部实体和进出系统的数据流都画在一张图中。

图 2-2 就是一个 Context 图的实例，只不过在绘制时做了一些处理，使得它看上去更加直观易懂。

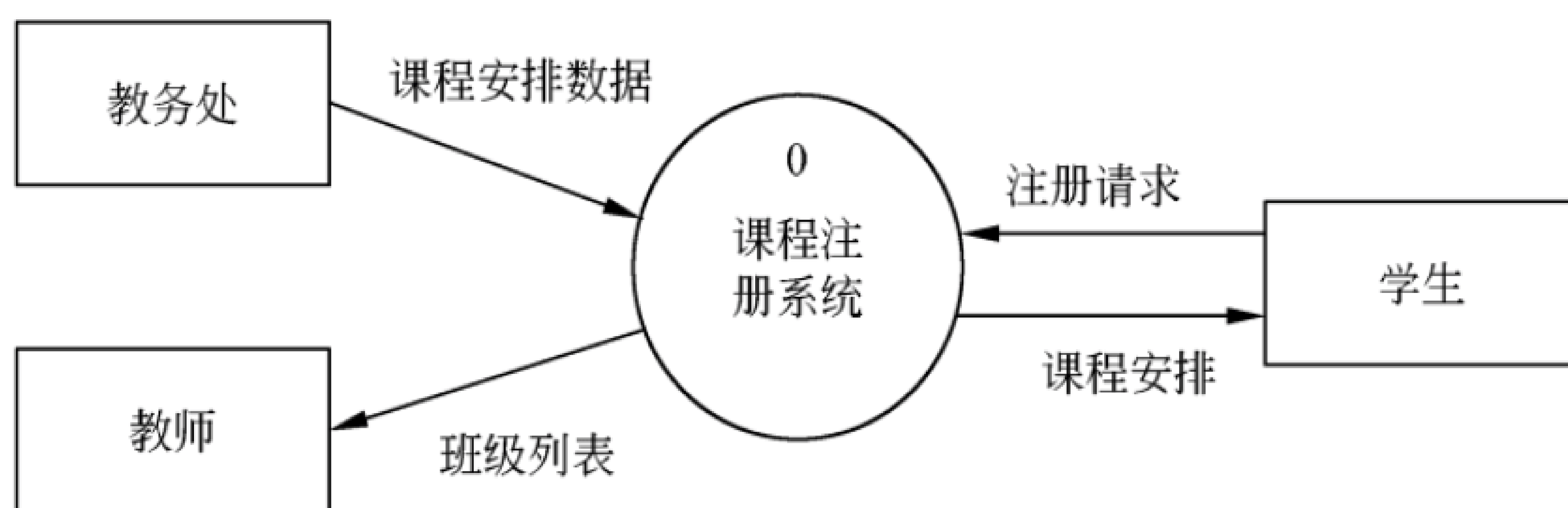


图 2-2　Context 图实例

Context 图用来描述系统有什么输入、输出数据流，与哪些外部实体直接相关，可以把整个系统的范围勾画出来。

（3）逐级分解。当完成了 Context 图的建模之后，就可以在此基础上进行进一步的分解。对图 2-2 进行再分解，在对原有流程了解的基础上，可以得到图 2-3。

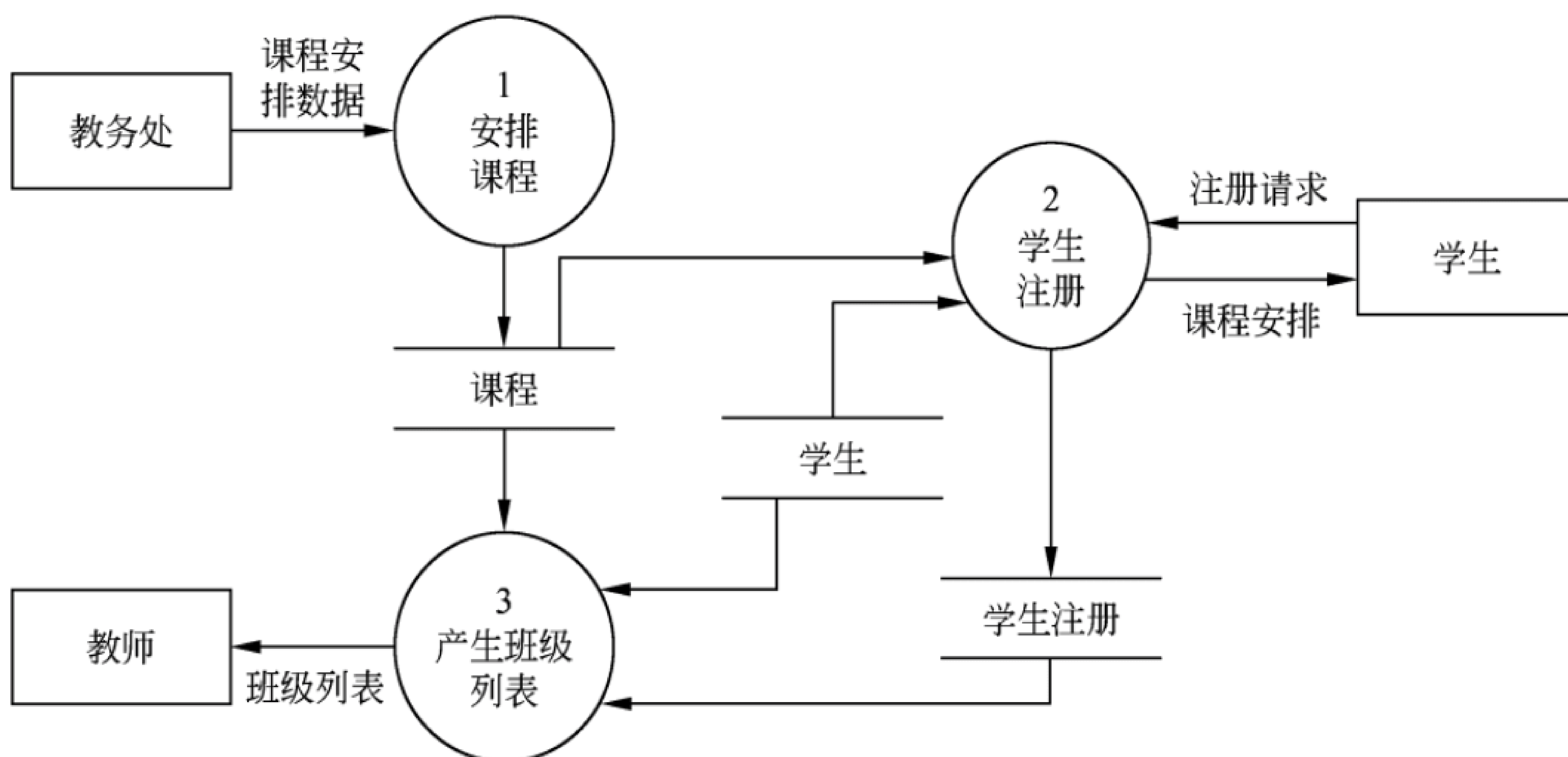


图 2-3　DFD 0 层图

图 2-3 是在 Context 图的基础上做的第一次分解，而在 Context 图只有一个加工，那

就是系统本身，可以将其编号为 0。而接下来对 Context 图进行的分解，其实就是对这个编号为 0 的过程进行更细化的描述，在这里引入了新的加工和数据存储，为了能够区分其位于的级别，在这个层次上的加工将以 1，2，3 为序列进行编号。

正是由于这是对加工 0 的分解，因此也称之为 DFD 0 层图。而可以根据需要对 DFD 0 层图上的过程（编号为 1，2，3）进行类似的再分解，称之为 DFD 1 层图，这 DFD 1 层图中引入的新过程，其编号规则就是 1.1，1.2…，以及 2.1，2.2…，依次类推，直到完成分析工作。

另外，这里存在着一个很关键的要点，那就是 DFD 0 层图是 Context 图的细化，因此，所有的输入和输出应该与 Context 图完全一致。这就是 DFD 的平衡原则。

（4）如何画 DFD。DFD 的绘制是一个自顶向下、由外到里的过程，通常按照以下几个步骤进行。

① 画系统的输入和输出：就是在图的边缘标出系统的输入、输出数据流。这一步其实是决定研究的内容和系统的范围。在画的时候，可以先将尽可能多的输入、输出画出来，然后再删除多余的，增加遗漏的。

② 画 DFD 的内部：将系统的输入、输出用一系列的处理连接起来，可以从输入数据流画向输出数据流，也可以从中间画出去。

③ 为每一个数据流命名：命名的好坏与 DFD 的可理解性密切相关，应避免使用空洞的名字。

④ 为加工命名：使用动宾短语为每个加工命名。

⑤ 不考虑初始化和终点，暂不考虑出错路径等细节，不画控制流和控制信息。

3. 细化记录 DFD 部件

为了能够更好地描述 DFD 的部件，SA 方法还引入了数据字典、结构化语言以及决策树和决策表等工具。通过使用这些工具，能够对 DFD 中描述不够清晰的地方进行有效的补充。

（1）结构化语言。结构化语言是结构化编程语言与自然语言的有机结合，可以采用顺序结构、分支结构、循环结构等机制，来说明加工的处理流程。该技术通常用来描述一些重要的、复杂的加工的程序逻辑。例如，下面就是一个使用结构化语言描述的例子。

```
IF 分数>=60 Then
   IF 分数<80 Then
      成绩=C
   ELSE
     IF 分数<90 Then
        成绩=B
     Else
        成绩=A
```

```
      EndIf
    EndIf
  ELSE
     IF 分数>=50 Then
        成绩=D
     ELSE
        成绩=E
     EndIf
  EndIf
```

（2）决策表和决策树。决策表是一种处理逻辑的表格表示方法，其中包括决策变量、决策变量值、参与者或公式。与上例相应的决策表如表 2-1 所示。

表 2-1　决策表示例

分数>=60	是			否	
分数	>=80		<80	>=50	<50
分数	>=90	<90			
成绩	A	B	C	D	E

决策树则使用像树枝一样的线条对过程逻辑进行图表化的描述，与表 2-1 相对应的决策树如图 2-4 所示。

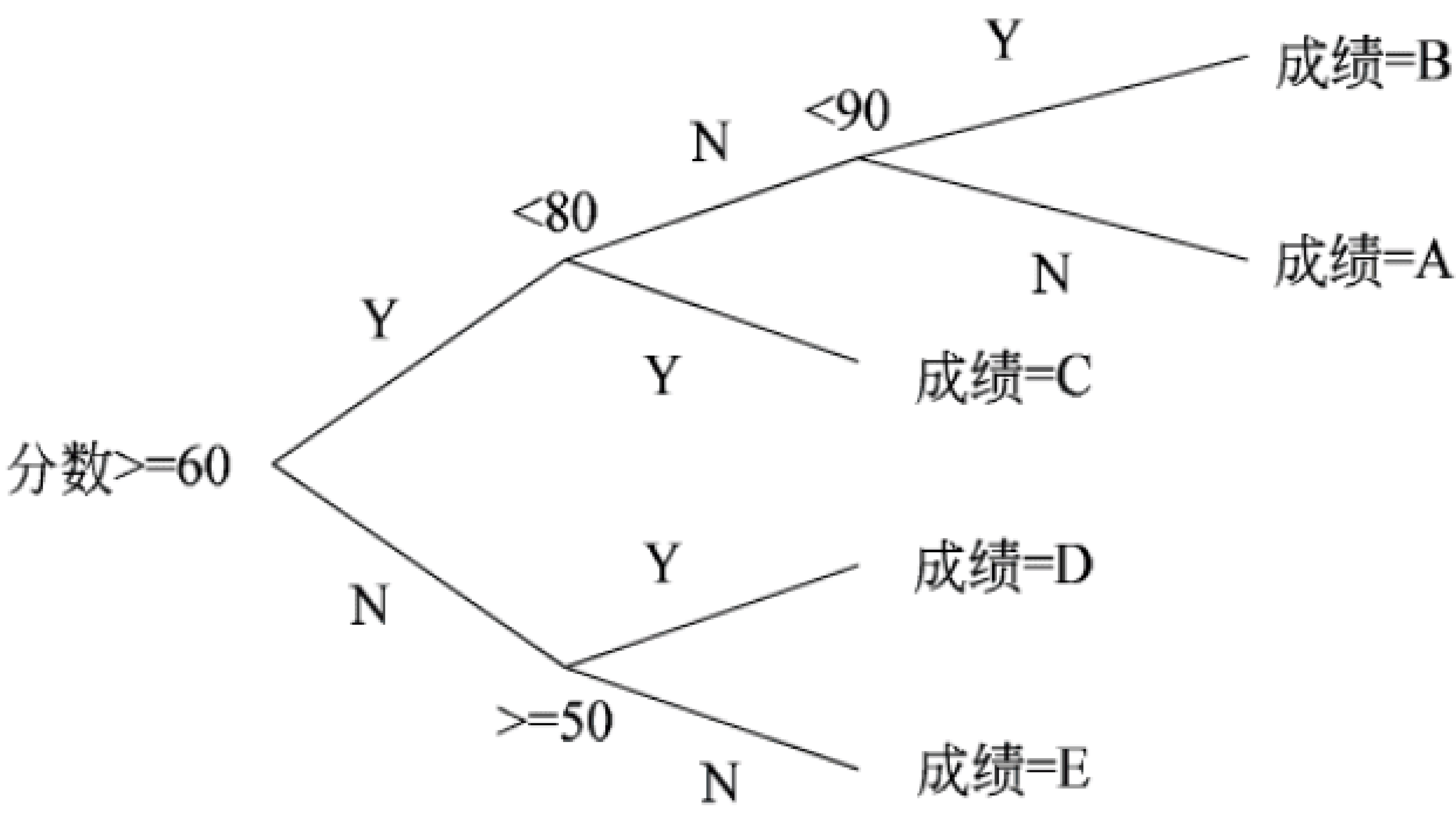


图 2-4　决策树示例

很显然，应用决策表和决策树来描述复杂决策逻辑，要远远优于使用结构化语言。而这两种技术也各有优劣，决策表更严密，决策树更易读。系统分析师可以根据自己的实际需要来灵活选择应用。

（3）数据字典。数据字典是一种很实用、有效的表达数据格式的技术，它是对所有与系统相关的数据元素的一个有组织的列表和精确的、严格的定义，使用户和系统分析师对于输入、输出、存储成分有共同的理解。通常，数据字典的每一条目中包括以下信息。

① 名称：数据或控制项、数据存储或外部实体的名称，如果有别名的还应该将别名列出来。

② 何处使用/如何使用：使用数据或控制项的加工列表，以及如何使用。

③ 内容描述：说明该条目内容组成，通常采用以下符号进行说明。

=：由…构成。

+：和，代表顺序连接的关系。

[|]：或，代表从中选择一个。

{}*：n 次重复。

( )：代表可选的数据项。

…：表示特定限制的注释。

④ 补充信息：关于数据类型、默认值、限制等信息。

下面就是一个数据字典的实例。

客户基本信息 = 客户编号+客户名称+身份证号码+手机+办公电话+家庭电话
客户编号 = $\{0...9\}^{8}$
客户名称 = $\{$汉字$\}^{4}$
身份证号码 = $[\{0...9\}^{15} | \{0...9\}^{18}]$
手机 = $[\{0...9\}^{11} | \{0...9\}^{12}]$
办公电话 =（区号）+本地号
家庭电话 =（区号）+本地号
　　区号 = $\{0...9\}^{4}$
　　本地号 = $[\{0...9\}^{7} | \{0...9\}^{8}]$

4. 实体-联系图

传统的系统开发方法都把重点集中在新系统的数据存储需求上，包括数据实体、属性，以及它们之间的关系。而描述这些东西的最好形式就是借助 E-R 图。E-R 图包括以下三个要素。

（1）实体：用矩形框表示，框内标注实体名称。

（2）属性：用椭圆形表示，并用连线与实体连接起来。

（3）实体之间的联系：用菱形框表示，框内标注联系名称，并用连线将菱形框分别与有关实体相连，并在连线上注明联系类型。

例如，图 2-5 就是一个教学系统的 E-R 图（为了简单起见，省略了部分实体的属性和联系的属性）。

有关 E-R 图的更详细的知识，请阅读本套丛书中的《系统分析师考试全程指导（2009 版）》的第 3.6.3 节。

5. 小结

SA 方法为开发者和客户提供了一个直观易懂的模型，能够实现对问题域的理解。

但是，SA 也存在着很多的先天不足：

（1）对问题域的研究力度不够大。

（2）分析与设计之间缺乏清晰的界限。

（3）没有一个真正的功能规格说明。

（4）需求实质上是根据满足该需求的某一特定系统的内部设计来加以说明的，而内部设计的开发则是使用不可靠的功能分解技术。

（5）不适用于很多类型的应用。

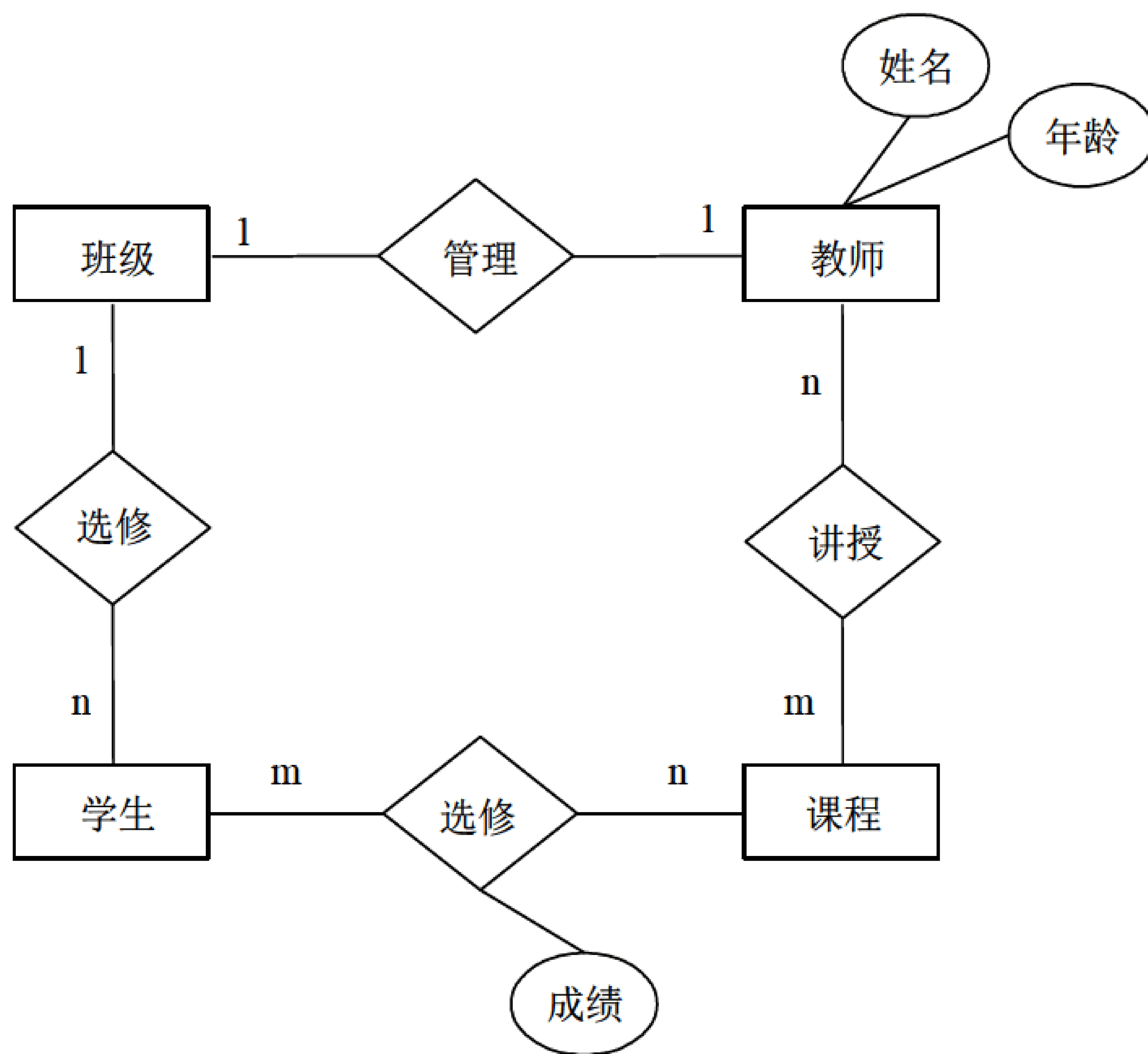


图 2-5　某教学系统 E-R 图

希赛教育专家提示：SA 方法在很大程度上推动了分析技术的发展，但又被更合适的技术逐渐取代。不过，在将来的某一天，SA 方法即使成为“明日黄花”，其具体工具仍然还会有很广泛的应用空间。

2.4.2　面向对象分析

在结构化的理论基础下，将应用分解成为一个个功能模块、子模块、功能接口等，它完全与现实问题域的东西没有具体的联系。而使用 OOA 的思想所建立的系统模型就是对问题域的完整、直接的映射，也就是从现实世界中抽象出一个模型，然后在计算机中实现。

从面向对象的角度来看，世界是由对象组成的。任何给出的业务功能都是由一整套协作的对象所支持实现的。因此，采用 OOA 方法时，需要识别出与系统相关的对象，并且描述这些对象的属性，以及它们之间的关系。同时，还需要了解这些对象之间是如

何协作，从而完成系统功能的。

总结而言，采用 OOA 方法，整个分析阶段通常包括以下两项工作任务：建立一个反映问题域静态关系的概念模型，通常使用类图来表示；建立一个反应系统行为的动态模型，即用例模型。

1. 建立域模型

问题域是指一个包含现实世界事物与概念的领域，这些事物和概念与所待开发的系统要解决的问题有关。而建立概念模型，又称为问题域建模、域建模，也就是找到代表那些事物与概念的对象。

1）寻找类

寻找类的方法有很多种，其中最广泛应用的莫过于“名词动词法”，也就是阅读需求文档，找出名词和名词短语，从中提取对象与属性。通常，带有所有格的名词是属性而非对象；找出动词与动词短语，从中提取操作与关联。

第一步，列出所有的备选类。也就是将需求中所有的名词和名词短语列举出来。

第二步，决定候选类。不是所有的名词和名词短语都是系统中的一个合适的候选类，因为有的在系统之外，有的与系统不相关，有的名词概念较小，只适合于作为某个候选类的属性。因此，必须对其进行一番筛选，把不合适的滤掉。

希赛教育专家提示：在采用名词动词法寻找类的时候，有些团队会陷入一个误区，那就是花费过多的时间于此，甚至到了“咬文嚼字”的地步，这会使得分析迷失方向。

2）确定类之间的关联

当完成了类的寻找工作之后，就需要理清这些类之间的层次关系，例如，关联、继承、聚合等。然后，通过 UML 的类图将这些关系记录下来。例如，图 2-6 就是与希赛教育图书管理系统相关的一个领域模型。

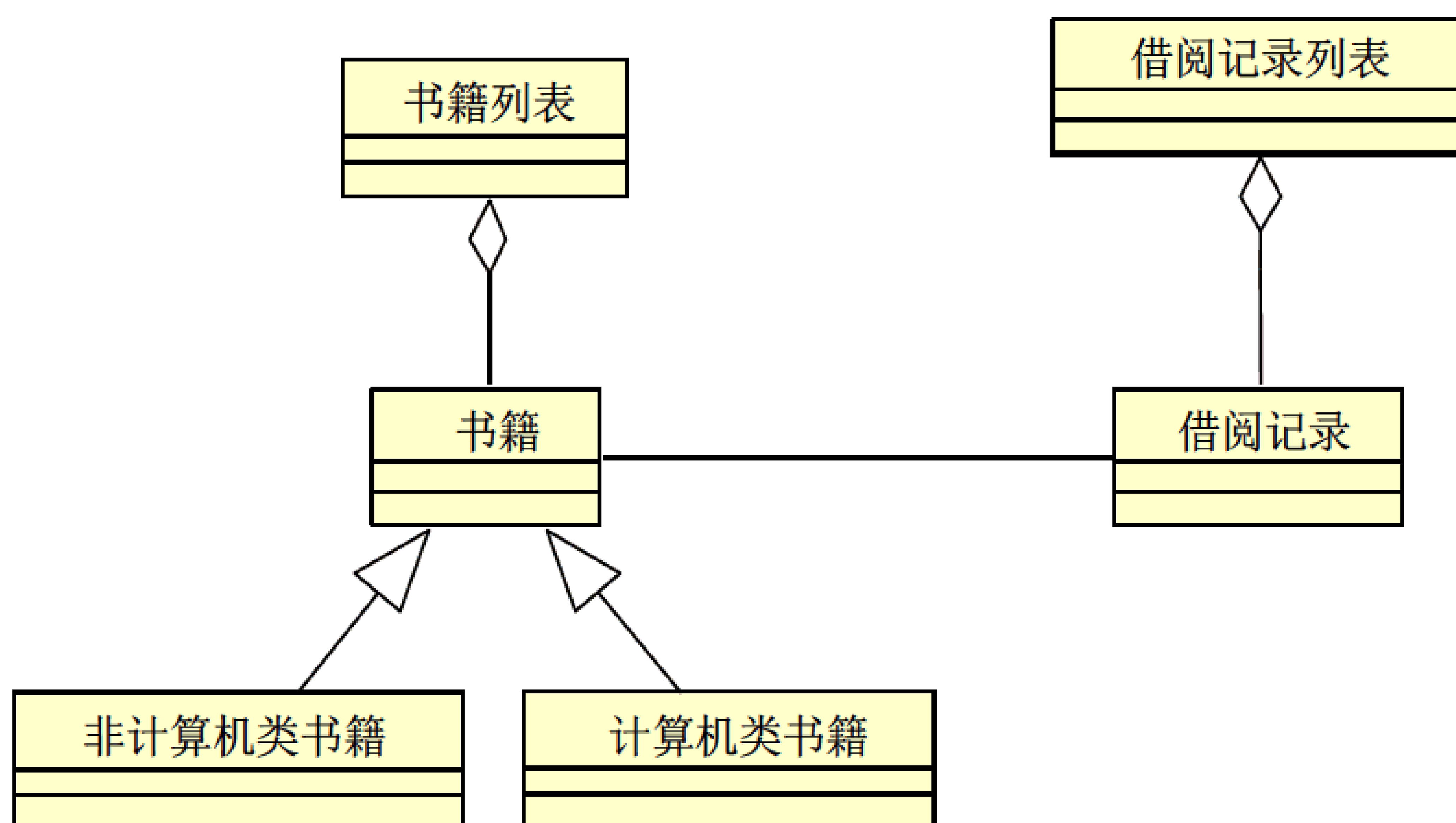


图 2-6　领域模型示例

当完成了这些关联关系的建模之后，就需要细化这些关系的描述。例如，从图 2-6

中就会产生一些疑问：一本书可以有多少条借阅记录等。那么，就需要将这些关系之间的多重性（多重度）进行描述。当然，这些描述必须是来源于业务规则，与领域相关的，现在还不清晰的可以暂时放在一边。图 2-7 就是一个修改过后的模型。

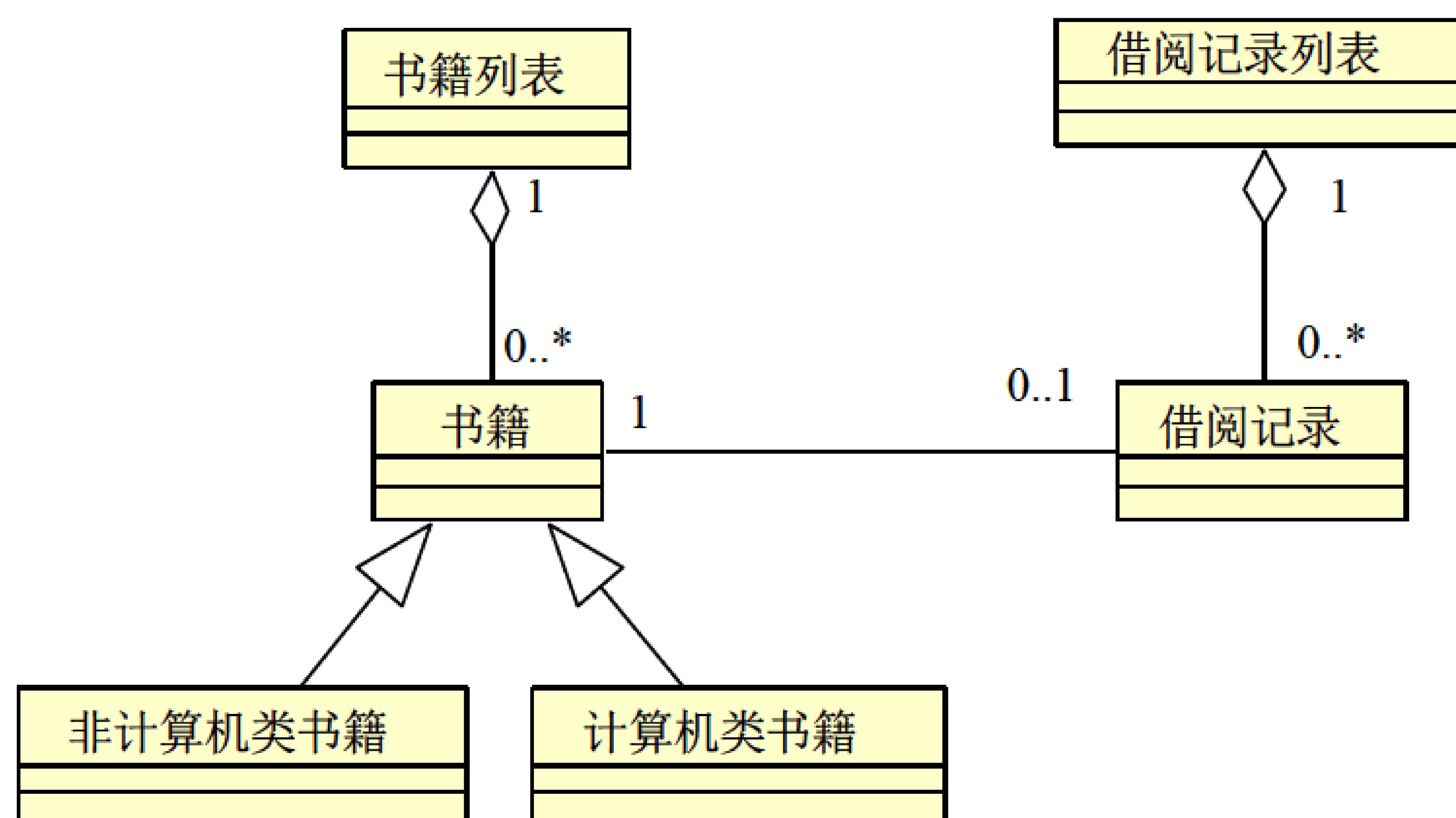


图 2-7　修改后的域模型

通过关联关系的识别与建模，就可以对问题域中的各个类之间层次结构关系、协作关系有一个相对完整的认识与理解。

3）为类添加职责

在前面两步中，找到了与问题域本质相关的主要概念类，而且还理清了它们之间的协作关系。接下来，就应该为这些类添加其相应的主要职责。类的职责包括两个方面的内容，一个是类所维护的知识，即成员变量、属性；另一个是类能够执行的行为，即成员方法。要注意的是，为类添加职责与找到类之间的关联关系一样，这个阶段也只能找到那些主要的、明显的、与业务规则相关的部分。切忌在这个阶段不断地细化，甚至引入了一些与具体实现相关的技术内容（例如，数据库、分布式对象之类的东西）。图 2-8 是一个完成了主要类职责之后的概念模型。

4）域模型的详细度

有些文献中建议只列出类以及类之间的主要关联关系，不要对关联关系进行描述，更不要体现类的职责；而有些文献则认为应该将这些东西都列出来。其实，干巴巴地讨论这个问题是没有任何意义的，根据笔者长期的实践，总结了以下两点：

（1）概念模型的目的是让开发团队对问题域建立一个全貌式的了解，并作为今后进一步深化建立基础，因此，不妨取中庸之道，将需求描述中所谈到的主要内容都反馈到概念模型中去，而无需顾及其是关联关系还是职责描述。

（2）概念模型是在开发过程中生产出来的第一个系统的静态模型，随着开发活动的推进，该模型将会随之加入新内容，改去旧内容，逐渐完善，演变完整。

总之，模型不是要生产的目标产物，而是过程中的一个辅助工作，只要能够利用其

帮助团队更好的开发，详细也罢，简约也罢，都是好模型。

2. 建立用例模型

有些制作精细的模型车不管从外观上还是内部结构上都与真车一模一样，但是却不能够像真车那样行驶，缺了什么呢？缺的是每个零件只是“神似”，而非“真是”。换一句话说，就是处于静态状态下是相像的，但是无法动起来，无法实现这些零件本该实现的功能，这也就使得模型车无法真正地开起来。

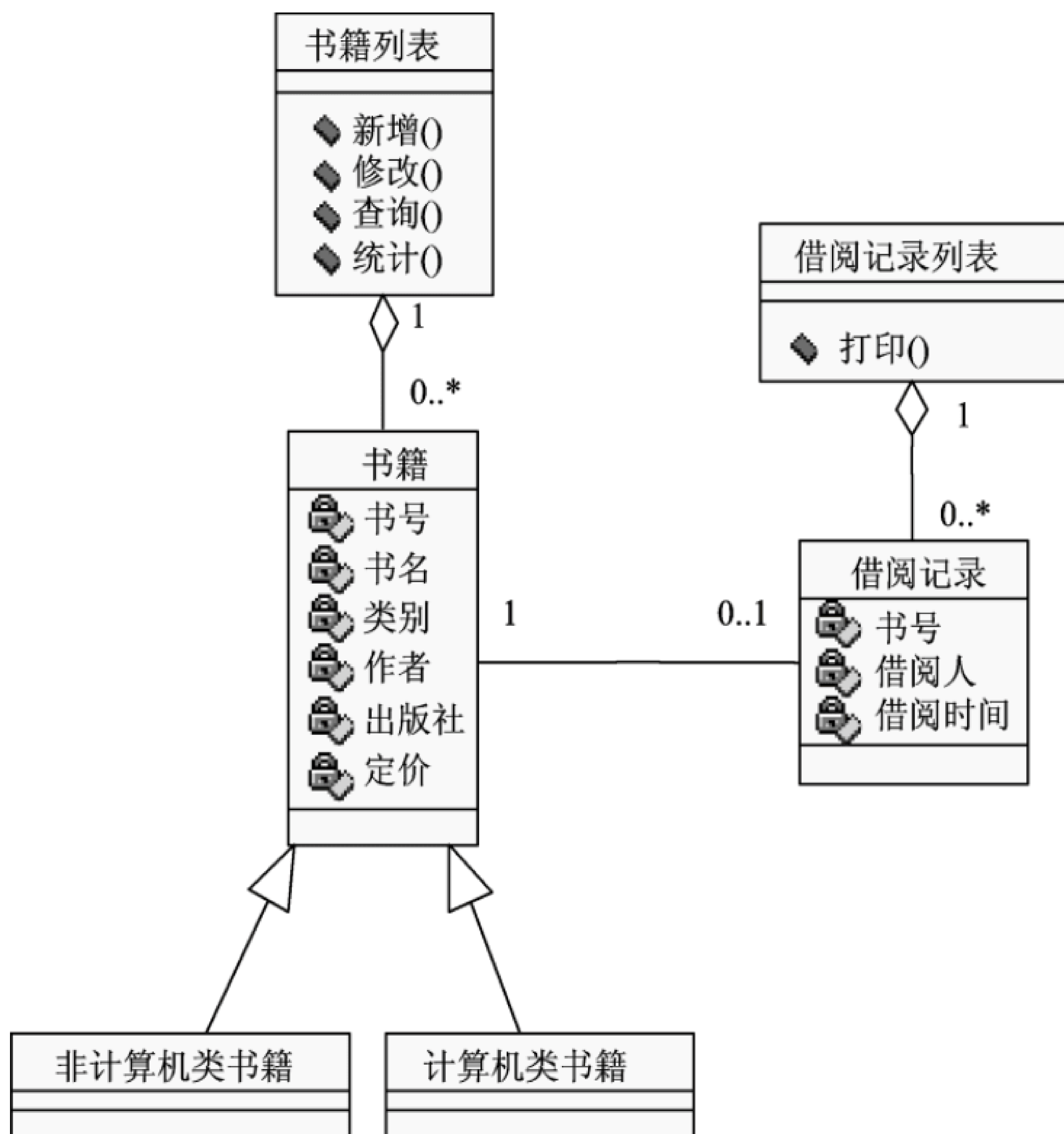


图 2-8　完整的域模型示例

当完成了概念模型的建立时，就是仅仅打造完成了一辆“模型车”而已，只能够帮助开发团队更好地理解系统所涉及的问题领域，对要开发系统相关的业务知识建立正确、完整、清晰的理解，但还无法开始构建系统。要想让“模型车”开起来，最重要的就是建立反映系统行为的动态模型，也就是用例模型。

（1）用例的概念。Ivar Jacobson 是这样定义用例的：用例实例是在系统中执行的一系列动作，这些动作将生成特定参与者可见的价值结果。一个用例定义一组用例实例。首先，从定义中得知，用例是由一组用例实例组成的，用例实例也就是常说的“使用场景”，就是用户使用系统的一个实际的、特定的场景；其次，用例应该给参与者带来可见的价值，这点很关键；最后，用例是在系统中的。

（2）用例模型的产生。用例技术自从诞生之日起，就被广为关注，现在已经成为了

现代软件工程公认的最佳需求分析实践之一。近几年来，在国内的开发团队中也开始逐渐被接受，不过由于国内认识用例是从 UML 开始的，就使得许多人误把用例图当作用例模型。另外，还有一些刚刚开始使用用例分析的开发组织，误将其当做一种分解技术，致使做出来的分析与功能分解酷似，以致失去了用例分析技术带来的益处。

其实，用例分析技术是一种需求合成技术，它的合成过程如图 2-9 所示。也就是采用现有的需求获取技术从客户、原有系统、文档中找到需求，记录下来，然后从这个零散的要求、特性中进行整理、提炼，从而建立用例模型。

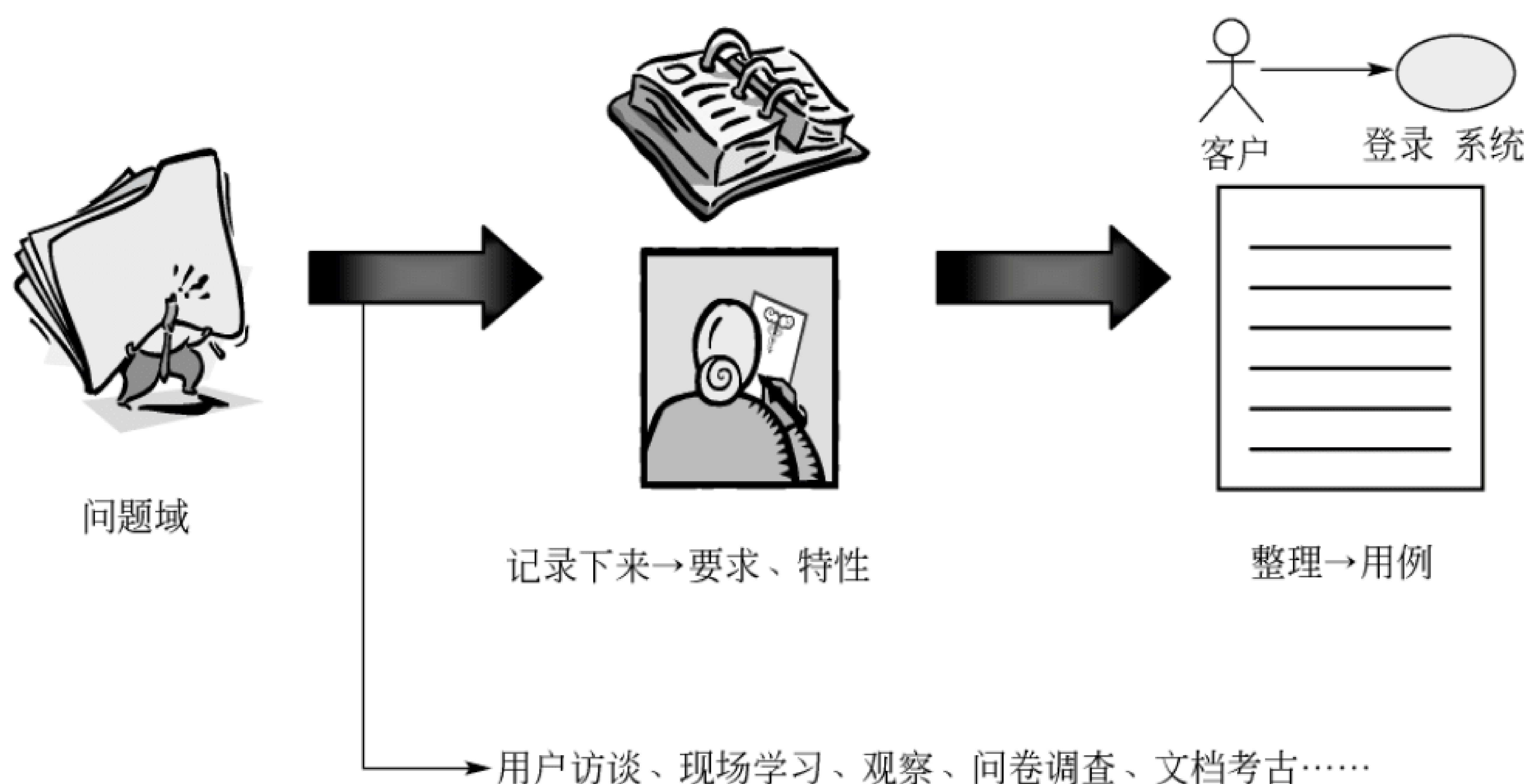


图 2-9 用例建模过程示意图

（3）识别参与者。参与者是与系统交互的所有事物，该角色不仅可以由人承担，还可以是其他系统、硬件设备，甚至是时钟。

① 其他系统：当系统需要与其他系统交互时，其他系统就是一个参与者。例如，在开发希赛教育图书管理系统时，希赛教育网站的图书系统就是一个参与者。

② 硬件设备：如果系统需要与硬件设备交互时，硬件设备就是一个参与者。例如，在开发 IC（Integrated Circuit，集成电路）卡门禁系统时，IC 卡读写器就是一个参与者。

③ 时钟：当系统需要定时触发时，时钟就是一个参与者。例如，在开发希赛教育网在线测试系统中的“定时交卷”功能时，就需要引入时钟作为参与者。

要注意的是，参与者一定在系统之外，不是系统的一部分。可以通过下列问题来帮助发现系统的参与者：谁使用这个系统？谁安装这个系统？谁启动这个系统？谁维护这个系统？谁关闭这个系统？哪些(其他的)系统使用这个系统？谁从这个系统获取信息？谁为这个系统提供信息？是否有事情自动在预计的时间发生？

（4）合并需求获得用例。将参与者都找到之后，接下来就是仔细地检查参与者，为每一个参与者确定用例。而其中的依据主要可以来源于已经获取得到的“特征表”，将特

征分配给相应的参与者。首先，要将这些获取到的特征，分配给与其相关的参与者，以便可以针对每个参与者进行工作，而无遗漏；其次，进行合并操作。在合并之前，首先还要明确为什么要合并，知道了合并的目的，才可能选择正确的合并操作。一个用例就是一个对参与者来说可见的价值结果，因此，合并的根据就是使得其能够组合成为一个可见的价值结果。合并后，将产生用例，而用例的命名应该注意采用“动词（短语）+名词（短语）”的形式，而且最好能够对其进行编号，这也是实现跟踪管理的重要技巧，通过编号可以将用户的需求落实到特定的用例中去。

（5）绘制成用例图。将识别到的参与者和合并生成的用例，通过用例图的形式整理出来，以获得用例模型的框架，也算是得到一个中间的成果，如图 2-10 所示。

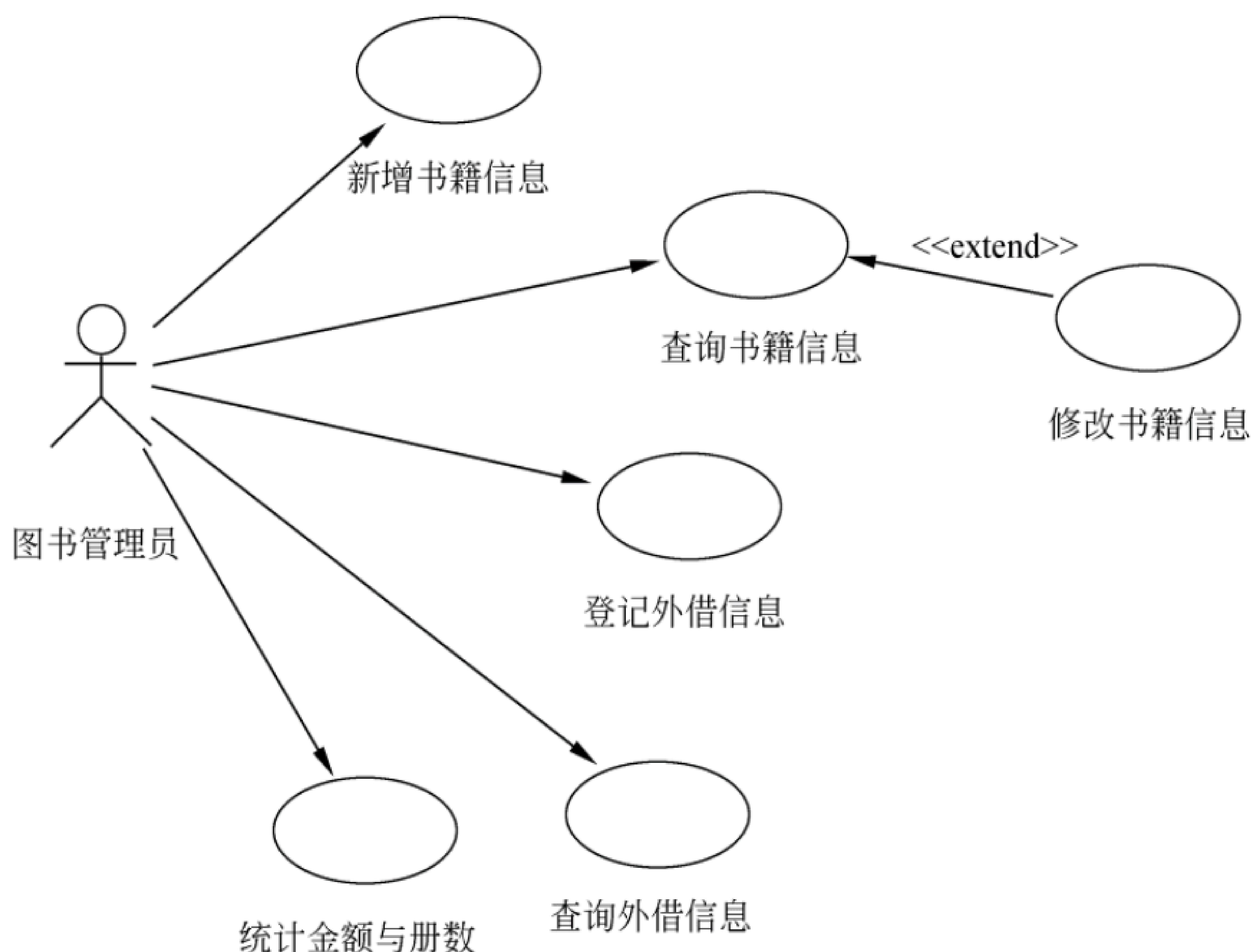


图 2-10　用例图示例

千万不要以为到此，用例分析就结束了。这仅仅是一个好的开端，接下来的工作才是最为重要的一环，也是用例发挥作用的关键。

（6）细化用例描述。用例的描述可以迭代完成，先对一些重要的用例编制相对细致的用例描述，对于一些不重要的，可以留待以后再补充完成。用例描述通常包括以下几个部分。

① 用例名称：应该与用例图相符，并写上其相应的编号。

② 简要说明：对该用例对参与者所传递的价值结果进行描述，应注意语言简要，使用用户能够阅读的自然语言。

③ 事件流：也就是该用例所完成的工作步骤，在编写时应该注意以下问题。

- 使用简单的语法：主语明确，语义易于理解。
- 明确写出“谁控制球”：也就是在事件流描述中，让读者直观地了解是参与者在控制，还是系统在控制。
- 从俯视的角度来编写：指出参与者的动作，以及系统的响应，也就是第三者的角度。
- 显示过程向前推移：也就是每一步都有前进的感觉。
- 显示参与者的意图而非动作（单有动作，让人不容易直接从事件流中理解用例）。
- 包括“合理的活动集”（带数据的请求、系统确认、更改内部、返回结果）。
- 用“确认”而非“检查是否”，例如，系统确认所输入的信息中书名没有重名等。
- 可选择地提及时间限制。

另外，事件流的编写过程也是可以分阶段、迭代进行的，对于优先级高的用例花更多的时间，更加的细化；对优先级低的用例可以先简略地将主要事件流描述清楚，次要事件流留待以后处理。对于一些事件流较为复杂的，可以在用例描述中引用顺序图、状态图、通信图等手段进行描述。

④ 非功能需求：主要对该用例所涉及的非功能性需求进行描述。由于其通常很难在事件流中进行表述，因此单列为一小节进行阐述。这些需求通过包括法律法规、应用程序标准、质量属性（可用性、可靠性、性能、支持性等）、兼容性、可移植性，以及设计约束等方面的需求。在这些需求的描述方面，一定要注意使其可度量、可验证。否则，就容易流于形式，形同摆设。

⑤ 前置条件：是执行用例之前必须存在的系统状态，这部分内容如果在当前阶段不容易确定，则可以在以后再细化。

⑥ 后置条件：用例执行完毕系统可能处于的一组状态，这部分内容如果在当前阶段不容易确定，则也可以在以后再细化。

⑦ 扩展点：如果包括扩展（或包含）用例，则写出扩展（或包含）用例名，并说明在什么情况下使用。

⑧ 优先级：说明用户对该用例的期望值，可以为今后开发时制定先后顺序。可以采用满意度/不满意度指标进行说明。

下面是图 2-10 中一个较重要的用例“新增书籍信息”的用例描述，这里给出的是一个相对完整的版本，这些内容不一定要一次完成。

1.用例名称

  新增书籍信息（UC01）

2.简要说明

  录入新购书籍信息，并自动存储建档。

3.事件流

  3.1 基本事件流

1）图书管理员向系统发出"新增书籍信息"请求；

2）系统要求图书管理员选择要新增的书籍是计算机类还是非计算机类；

3）图书管理员做出选择后，显示相应界面，让图书管理员输入信息，并自动根据书号规则生成书号；

4）图书管理员输入书籍的相关信息，包括：书名、作者、出版社、ISBN 号、开本、页数、定价、是否有 CDROM；

5）系统确认输入的信息中书名未有重名；

6）系统将所输入的信息存储建档。

3.2 扩展事件流

- 如果输入的书名有重名现象，则显示出重名的书籍，并要求图书管理员选择修改书名或取消输入；
- 图书管理员选择取消输入，则结束用例，不做存储建档工作；
- 图书管理员选择修改书名后，转到 5）

4.非功能需求

无特殊要求。

5.前置条件

用户进入图书管理系统。

6.后置条件

完成新书信息的存储建档。

7.扩展点

无

8.优先级

最高（满意度 5，不满意度 5）

3. 其他相关建模技术

除了前面提到了类图、用例图外，在面向对象的分析阶段，常用的图还包括顺序图、通信图和状态图。

（1）顺序图（Sequence Diagram）：是一种交互图（Interaction Diagram），交互图展现了一种交互，它由一组对象或角色以及它们之间可能发送的消息构成。交互图专注于系统的动态视图。顺序图是强调消息的时间次序的交互图。

（2）通信图（Communication Diagram）：也是一种交互图，它强调收发消息的对象或角色的结构组织。顺序图和通信图表达了类似的基本概念，但每种图所强调的概念不同，顺序图强调的是时序，通信图则强调消息流经的数据结构。

（3）状态图（State Diagram）：描述一个状态机，它由状态、转移、事件和活动组成。状态图给出了对象的动态视图。它对于接口、类或协作的行为建模尤为重要，而且它强调事件导致的对象行为，这非常有助于对反应式系统建模。

有关以上三种图的更详细的内容，请阅读本套丛书中的《系统分析师技术指南（2009版）》的第 13.5 节和第 13.6 节。

2.4.3 面向问题域的分析

相对来说，PDOA 是一项很新的技术，还处于研究阶段，相关的文档资料还不多。与 SA 和 OOA 相比，PDOA 更多地强调描述，而少强调建模。它的描述大致分为以下两个部分。

（1）关注问题域：用一个文档对含有的问题域进行相关的描述，并列出需在该域中求解的问题列表，也就是需求列表。只有这个文档是在分析时产生的。

（2）关注解系统的待求行为：用一个文档对解系统（即系统实现）的待求行为进行描述。该文档将在需求规格说明时再完成。

在 PDOA 方法中，对整个过程有着一个清晰的定义：

（1）搜集基本的信息并开发问题框架，以建立问题域的类型。

（2）在问题框架类型的指导下，进一步搜集详细信息，并给出一个问题域相关特性的描述。

（3）基于以上两点，收集并用文档说明新系统的需求。

从上面的描述中可以看出，问题框架是 PDOA 的核心元素，是将问题域建模成为一系列相互关联的子域，而一个子域可以是那些可能算是精选出来的问题域的一部分。也可以把问题框架视作是开发 Context 图，但不同的是，Context 图的建模对象是针对解系统，而问题框架则是针对问题域。也就是说，问题框架的目标就是大量地获取更多有关问题域的信息。

2.4.4 方法论的比较

SA 方法和 OOA 方法有着截然不同的思维方式。SA 方法关注于功能的分层和分解，这非常符合人们自上而下、逐步分解问题直到可解决的自然思考方式。SA 方法本身隐含着几个基本假设，即问题域是可定义的、问题域是有限的、通过有限的步骤总可以将复杂问题分解到可解决的程度。SA 方法应用的是科学方法中的因果律、归纳法和逻辑法，通过对现实世界中的问题域进行不断的“测量”和“分解”，直到得到问题域的逻辑模型。

OOA 方法则遵循完全不同的思维方法。OOA 方法基于抽象、信息隐藏、功能独立和模块化这些基本理念对系统进行分析。OOA 方法首先对问题域的事物的“外在表象”进行观测，然后在逻辑世界中“模拟”出一个对应的逻辑对象，“断定”该对象和现实事物是一致的。随后，观测到的对象被记录入对象集合，观测到的行为和表象被记录入对象–关系模型和对象–行为模型。OOA 方法建立的对象彼此之间通过接口来相互沟通，每传递一个消息即触发一个事件，并引起内部方法的执行。只有观测对象内部的时候，才能看到具体的属性和方法。否则，只能看到对象对外部开放的接口。

SA 方法假定系统分析师理解问题域的全部，并且有能力正确地识别和分解问题。

而 OOA 方法既不假定系统分析师理解问题域的全部，也不假定系统分析师能够建立正确的抽象对象，它只承诺一种可以持续“观测并理解”的方法，以及“观测后建立”的对象和现实世界的外在表象是一致的。

很难对 SA 方法和 OOA 方法作一种优劣性的比较，使用两种方法成功和失败的软件系统都很多，并且这两种方法也只是人类用于系统分析的方法论之二而不是全部。OOA 方法已经成为当前的主流分析方法，拥有大量的语言和建模工具的支持。然而，结构化方法也并未过时，很多成功的软件系统依然在通过结构化方法进行分析和实现。

下面，将比较方法论的立足点，放在“如何在实践中正确运用方法论”的认识上。毕竟，恰当地运用方法解决问题才是最重要的。

1. 关于结构化分析方法

使用 SA 方法可能存在的风险在于：在得到问题本质性的描述之前，不断分解出的结果和需要处理的信息越来越多，越来越复杂，使得“只见树木不见森林”的风险大大增加。因此，使用 SA 方法的人需要不断将得到的分析结果与现有的知识和问题解决体系进行比较、匹配和归结，从而保证能在细化问题的连续过程中，仍然能够把握问题的本质。

SA 方法运用越好，越要求系统分析师具有一种“俯视全局”的能力，能够透过问题的表象直接把握问题的本质。当系统分析师能力足够强、软件项目较小时，SA 方法有望快速找到最简洁、高效率的逻辑模型。但当项目复杂程度高和规模大的时候，SA 方法并不承诺提供控制错误蔓延的手段。而且，依赖系统分析师的个人能力解决复杂工程问题的程度毕竟是有限的。

2. 关于面向对象分析方法

OOA 方法提供了一个方便的、可持续观测和扩展系统的机制，也承诺抽象出的对象是健壮的、可控制的和容易维护的。由于 OOA 方法在“观测并模拟”的过程中，总是试图在问题中抽象出更公用的解释（类），因此，对系统分析师抽象事物和把握最初分析方向的要求很高，掌握难度要高于 SA 方法。

由于 OOA 方法通过“信息隐藏”的“封装”手段屏蔽了对象内部的执行细节。虽然控制了错误的蔓延，但却导致一旦错误发生，从系统行为定位故障所在的代价较大，尤其是那些“继承”程度过深的对象，可能导致理解问题的开销超过了解决问题的开销。这是使用 OOA 方法的代价。

对于那些经验不够丰富的系统分析师来说，常常难以控制抽象对象的层次、粒度，甚至可能抽象出与问题本质面目全非的对象模型来，这样的模型既不正确也难以进一步细化设计，反而增加了未来的修改难度。而且，这种体现在不正确的对象接口定义上的“预设观念”，常常会误导后续的改进者理解系统设计。因此，使用 OOA 方法时，正确抽象出软件系统的基础对象集合极其重要。否则，一旦后续的系统建构在一个错误划分的基础对象群体和对象继承关系上，可能造成灾难性的后果。原有的开发成果多半会被

全面抛弃，而不能修补再用。

对于需求变化频繁的系统，跟踪项目目标的变化是一个让人头疼的问题。而面向对象方法由于抽象层次比较高，通常增加了初期分析和设计的困难程度和开销，得到一个高度可复用的面向对象软件系统设计是一件困难的事情。

SA 方法和 OOA 方法不是对立的，或者说某种方法先进而另一种方法过时。在实际项目中常可以结合使用。例如，利用 SA 方法分析问题的信息加工域或归结具有共性的处理时，常常有助于确认类的划分规模、内部操作和接口。当问题领域比较复杂的时候，SA 方法能够帮助使用 OOA 方法的系统分析师确认系统最初的高阶模型，并指导后续的“观测”应该从何处开始。

下列原则可以用于指导如何运用这些分析方法：

（1）应关注运用这些方法的成本和价值。

（2）并不存在方法学上单纯的优劣和对立问题。

（3）在适当的环境适当地运用这些分析方法抓住问题的本质才是关键。

这些都需要系统分析师具有高度的思维水平，将知识、技能、经验、方法等高度科学、艺术性地运用在目标问题的解决上。

本章参考文献

[1] Karl E.Wiegers. 软件需求. 北京：机械工业出版社，2000

[2] Suznne Robertson, James Robertson. 掌握需求过程. 北京：人民邮电出版社，2003

[3] Ian K.Bray. 需求工程导引. 北京：人民邮电出版社，2003

[4] Soren Lauesen. 软件需求. 北京：电子工业出版社，2002

[5] Ian Sommerville. 软件工程. 北京：电子工业出版社，2003

[6] John W.Satzinger，Robert B Jackson，Stephen D Burd. 系统分析与设计. 北京：机械工业出版社，2002

第3章　系 统 设 计

系统设计是根据需求分析的结果，完成系统构建的过程。这是个令人兴奋，同时也是艰难而令人困惑的过程。在软件生命周期中，系统设计的好坏，对软件质量的影响，不亚于任何其他过程。设计师往往是在担负重任，也是许多人梦寐以求的工作。

3.1　系统设计概论

许多想进行系统分析与设计的年轻人以为自己知道了面向对象、统一建模语言、设计模式等新鲜深奥的名词就可以进行设计了，但是，掌握工具和技能决不是成为优秀的设计师的充分条件，甚至也不是必要条件。遗憾的是，这里没有捷径，只有设计师在实践中不断学习和总结。而在实践中，系统设计与其说是在设计，不如说在选择和妥协。

1. 妥协和选择

妥协，就是在各个系统目标之间找到一个平衡点。系统目标包括但不限于功能、性能、健壮性、开发周期、交付日期等。不幸的是，这些目标之间往往是矛盾的，提高软件性能直接意味着开发周期的增加、交付日期的推迟，盲目增加功能可能导致性能降低，维护成本提高。系统设计师的难题在于，如何在如此众多的目标之间找到一个平衡点，并且明确地知道如何设计能实现这个平衡，既可以把成本控制在预算之内，又能让用户相对满意。

可行性研究阶段应该已经论述了存在这样一个平衡点，但是，如果设计师惊恐地发现没有这样一个平衡点，如同没有一个设计师能让人骑着自行车到月球上去，那么设计师只能提出放弃某个方面的过度要求。否则，系统要遭受必然失败的命运。更多的情况是没有经验的设计师不知道是否存在这些平衡点，更不知道可以如何利用合理的设计以及有效的工具来达到平衡。因此，系统设计师需要了解可以解决问题的各种方案，并清楚地知道各个方案的效果、成本、缺点，以及这些方案的区别。在各种方案中进行选择。而这些，不是设计师能在短时间内就能学会的。

2. 不断的学习实践

系统设计师很少会完全重新设计一个新系统，他们总参考多个与目标系统相类似的系统，再从中进行甄别、取舍和补充来作为新系统的设计。人们发现一个优秀的设计师似乎能在听完需求后就立即构想出目标系统的框架，这并不是因为他聪明或者比初次茫然不知所措的设计师新手多一个脑袋，而是因为他们能够把以前的设计根据需要再次使用上，他在平时已经了解大量的系统，对各种设计的缺点、局限也了然于胸。所以，要

成为优秀的系统设计师，了解、掌握、消化、总结前人和自己以前设计成果是提高的最好方法，似乎也是唯一的方法。

系统设计师的苦恼有时候和程序员一样，计算机系统的发展是如此的快，解决方案也越来越多，如同编程语言的发展。同时，基于人类无止境的贪婪本性，老板和客户也提出了越来越高的要求，这又需要设计师不断学习、创造新的方案。

3. 规律和方法

系统设计也并非没有规律可以遵循，如同“幸福的家庭都很相似，不幸的家庭各有各的不幸”一样，人们在实践中发现优秀的系统设计一般在以下方面都很出色：

（1）组件的独立性。审视自己设计的系统，是否做到了高内聚，低耦合呢？

（2）例外的识别和处理。谁能保证系统使用者都精确按照使用说明书使用呢？

（3）防错和容错。当类似网络中断、数据库崩溃这样的灾难性事件发生时，系统也跟着崩溃吗？

而且，更幸运的是，也有一些技术能够改进系统设计，这些方法包括降低复杂性、通过和约进行设计、原型化设计、错误树分析等。

本章讨论系统设计中涉及的几个方面，读者可以发现这些内容几乎都可以写成一本书，要在短短的一章中将如此多的问题论述清楚，的确让笔者冥思苦想了很久。这里也许只能给出一个方向，并希望所举的实例能给读者一点点帮助。

3.2 处理流程设计

流程的设计是对设计师更高的挑战，现实中的对计算机所管理的流程需要灵活的定义、方便的路径修改、容易使用，可是这几个目标是矛盾的。更严重的是，如何分析现实中的流程本身就是个困难的问题，更不用谈如何来设计实现了。流程设计的主要困难实际上也就是软件的主要困难：现实复杂性。任何对现实的描述（图形也罢，文字也罢）都是不完美的，“不识庐山真面目”是所有设计师面临的共同难题。设计师不得不意识到所有的流程模型都是对现实的简化，计算机只根据确定的信息做判断，而现实中的流程存在大量的不确定性，虽然计算机专家们自信的告诉企业的管理者这是管理的问题，信誓旦旦的保证可以用计算机系统来“完善”企业的管理，但他们似乎没有意识到企业管理已经发展了几百年，而计算机还没有百年的历史。

人们常常抱怨计算机系统的流程设计太过刻板，因为许多时候，标准流程是先于应用构造的，且由一些集中的权威强制执行的，所以，这种刻板性是不可避免的。同时，对参与者而言，缺乏自由度导致工作流管理系统显得很不友好。结果是它们经常被忽略或绕过，甚至最终被放弃。另外的困难是，对于流程处理，不仅名称众多，例如，动态模型、工作流等，而且对流程的定义也是千姿百态。面对这些困难，系统设计师无疑需要巨大的勇气来进行流程设计。

3.2.1　一些基本概念

首先，介绍一些与系统处理流程设计（工作流设计）相关的基本概念。

（1）工作流。工作流管理联盟（Workflow Management Coalition，WFMC）对于工作流的定义如下：工作流是一类能够完全或者部分自动执行的经营过程，根据一系列过程规则、文档、信息或任务在不同的执行者之间传递、执行。简单地说，工作流是现实中的具体工作从开始到结束过程的抽象和概括。这个过程包括了众多因素，例如，任务顺序、路线规则、时间时限约束等。

（2）流程定义。流程定义是指对业务过程的形式化表示，它定义了过程运行中的活动和所涉及到的各种信息。这些信息包括过程的开始和完成条件、构成过程的活动以及进行活动间导航的规则、用户所需要完成的任务、可能被调用的应用、工作流引擎的引用关系，以及与工作流数据的定义。这个定义的过程可能是由设计师用纸和笔来完成，但越来越多的设计师倾向于使用流程定义工具来完成这个工作。

（3）流程实例。也常常称为工作，是一个流程定义的运行实例。例如，客户的一次定购过程，客服中心受理的一次客户投诉过程等。

（4）工作流管理系统。与数据库管理系统类似，是一个软件系统。这个程序存储流程的定义，按照所使用的流程定义来触发流程状态的改变，推动流程的运转。这个推动的依据常被称为工作流引擎。

（5）流程定义工具。也样是一套软件系统，这个软件和工作流管理系统的关系就如同数据库设计工具和数据库管理系统的关系一样。它可能是独立的软件，也可能是工作流管理系统的一部分。如前所述，设计师常常使用流程定义工具来完成流程定义的工作。它提供一些常用的工作流元素和素材，以提高设计师的效率。

（6）参与者。可以是具体的人或者角色（企业内部有特别共同作用的多个人），也可以是自动化系统，甚至是其他系统。

（7）活动。活动是流程定义中的一个元素，一次活动可能改变流程处理数据的内容、流程的状态，并可能将流程推动到其他活动中去。活动可以由人来完成，也可以是系统自动的处理过程，典型的自动处理是当活动超过了其可以容忍的时限时，自动过程将向流程定义中指定的参与者发出一条消息。

（8）活动所有者。参与者之一，他有权决定该活动是否结束，当他决定活动结束时，将活动推动到其他活动中，可能是下一个活动，也可能是前一个活动。

（9）工作所有者。是有权整体控制流程实例执行过程的参与者。

（10）工作项。代表流程实例中活动的参与者将要执行的工作。

要分析现实中的处理流程，必须首先描述目标系统的流程，这个过程也可以称为建模。不过“建模”这个词用得太烂，有人总喜欢用建模这个词来表示自己的高深，笔者还是喜欢“描述”这个词。发现流程是个复杂的事务，必须从多方面才可以描述一个流

程，包括流程的参与者、参与者做什么工作、工作完成的时间限制，还需要说明工作的数据流和完成工作的控制流。

人们认为自然语言在描述如此复杂的事务容易引起歧义，所以，定义了一些形式化语言，试图在自然语言中挑选一个子集，这个子集既可以真正描述流程，又能够摆脱自然语言的复杂和多变。实际上，想在人和机器在理解处理流程上架起一座桥梁，如同其他计算机语言和后来发展的 UML 一样，这些形式化的语言也称为工作流定义语言。人们希望这些不同的定义语言能和 SQL（Structured Query Language，结构化查询语言）一样，最终有一个统一的标准。

同样，为了描述实际中的处理流程，人们也想到了图形的方式。有限状态自动机是一种分析状态和改变状态的良好工具。这种方法需要完全列出流程中所有状态以及这些状态的组合，当处理流程变得庞大时，自动机所对应的状态图膨胀得太厉害。由于 Petri 网有严格的数学基础和图形化的规范语义，在描述离散事件动态系统方面的能力已经得到公认，具有较强的模型分析能力，在工作流的描述和分析中已经是人们广泛采用的一种方法。虽然有人认为，图形并非工作流的最佳表示方法，但由于图形的直观性，大多数设计师都愿意采用图形的描述方式。

3.2.2 工作流管理系统

根据 WFMC 的定义，工作流管理系统是“一种在工作流形式化表示的驱动下，通过软件的执行而完成工作流定义、管理及执行的系统”，其主要目标是对业务过程中各活动发生的先后次序，以及与活动相关的相应人力或信息资源的调用，进行管理而实现业务过程的自动化。

如同关系数据库一样，现在已经出现了专门的工作流管理系统，这些系统经过专门的设计，从不同的角度负责解决设计师在流程设计中遇到的共同问题：节点定义、路径选择、数据流动等。不幸的是，和关系数据库共同基于关系代数、支持标准的 SQL 不同，这些工作流管理系统基于不同的数学模型，提供完全不同的接口。所以，这些工作流管理系统各具特色，但在通用性和其他系统相互协作上的不足使得这些系统的应用受到了限制。

WFMC 给出了包含 6 个基本模块的参考模型，这 6 个模块被认为是工作流管理系统的最基本组成，这个参考模型同时包括了这些模块之间的接口标准。

（1）流程定义工具。这部分软件提供图形化或者其他方式的界面给设计师。由设计师将实际工作流程进行抽象，并将设计师提交的流程定义转换为形式化语言描述，给工作流执行服务进行流程实例处理提供依据。

（2）工作流执行服务。这个服务程序是工作流管理系统的核心，它使用一种或者多种数据流引擎，对流程定义进行解释，激活有效的流程实例，推动流程实例在不同的活动中运转。与包括客户应用程序、其他工作流执行服务程序，以及其他应用程序进行交

互，从而完成流程实例的创建、执行和管理工作。同时，这部分软件为每个用户维护一个活动列表，告诉用户当前必须处理的任务。如果有必要，还可以通过电子邮件甚至是短消息的形式提醒用户任务的到达。

（3）其他工作流执行服务。大型的企业工作流应用，往往包括了多个工作流管理系统。这就需要这些工作流管理系统之间进行有效的交互，避免画地为牢，出现信息孤岛的现象。

（4）客户应用程序。这是给最终用户的界面，用户通过使用这部分软件对工作流的数据进行必要的处理，如果用户是当前活动的拥有者，还可通过客户应用程序改变流程实例的活动，将流程实例推动到另外一个活动中。

（5）被调用应用程序。这常常是对工作流所携带的数据的处理程序，用得很多的是电子文档的处理程序。它们由工作流执行服务在流程实例的执行过程中调用，向最终用户展示待处理数据。在流程定义中应该定义这些应用程序的信息，包括名称、调用方式、参数等。

（6）管理和监控工具。如同数据库管理系统或多或少地提供一些方式告诉管理员当前数据库的使用状态一样，工作流管理系统也应该提供对流程实例的状态查询、挂起、恢复、销毁等操作，同时提供系统参数、系统运行情况统计等数据。

读到这里，没有处理流程设计经验的设计师一定已经云里雾里了。确实，流程设计是系统设计中最困难的一部分，它的复杂性直接来源于现实世界的复杂性。而且，直到现在，人们对流程的设计，仍然是在探索之中。

3.3　系统文件设计

文件可以作为保存数据的方法，也可以作为不同计算机、不同系统之间的数据交换的方法。当然，为了这两个目的，现在都有很多其他方法，不一定需要文件的方式。同时，随着计算机的发展、速度的提高、软件的增强，即使是桌面数据库都已经有了相当强大的功能，所以，只要条件允许，设计师大多数会采用数据库作为数据的保存方式，而且关系数据库已经非常成熟，理论和实践都有了相当的进展，工具异常丰富，实现也变得非常简单。尽管如此，作为数据保存，文件方式仍然还有一些优势，主要适用于以下方面：

（1）数据量大，但数据并不是数字、字符等结构化数据。例如，多媒体信息等。

（2）数据量大，信息松散。例如，历史记录、档案文件等。

（3）临时存放的数据。

（4）非关系数据、层次化数据。例如，系统配置信息等。

3.3.1 文件逻辑结构

从文件的逻辑结构来看，文件的分类有多种方式。

（1）文本文件和非文本文件。按照是否可以阅读，分为文本文件和非文本文件，文本文件可以直接提供阅读，同时占有了比较大的存储空间；非文本文件只有专门的程序才能够阅读，一般是只给计算机读的资料，可以节省空间，处理速度也快一点。由于计算机的发展，存储空间和处理能力都有了极大的提升，价格不断下降，所以，除非特殊原因（如保密等），设计师倾向于选择文本文件，这样，在实现和调试排错的过程中，文件的内容一目了然，能够节约开发成本。

（2）流式文件和非流式文件。流式文件经常应用于影音数据的存储，特点是即使文件数据不完整，也可以展示其存在的部分；非流式文件必须有完整的文件才能理解文件的内容。显然，对于非影音数据，很难用流式文件来保存，设计师需要采用其他一些技术（如校验等）来保证文件的完整性。

（3）压缩文件和非压缩文件。压缩算法已经有很多，随着 Internet 的兴起，ZIP 的压缩格式普遍流传，几乎所有的压缩工具都支持 ZIP 的压缩格式。一般在文件副本的保存、文件的传递、存储空间受到限制时，设计师会采用压缩文件。

3.3.2 文件物理结构

按照存储组织结构，文件可以分为流水文件和顺序文件。

1. 流水文件

流水文件又称为堆文件或流年文件。这种文件只是简单地按照数据到达的时间顺序依次存储数据。这些数据的格式可以相同，也可以不同。显然，这种处理方式的文件结构简单，有很快的插入速度，但是查找信息的速度却很慢，修改和删除也不方便。因此，流水文件最常用于历史日志的记录，也可用于待处理数据的暂时存放。

2. 顺序文件

顺序文件是为了快速响应而专门设计的。顺序文件最重要的用途是用来构造数据库。它使用确定的关键字作为数据存储位置的依据。顺序文件有多种存储结构形式，主要的有向量结构、链结构、块链结构和 Hash 文件。

向量结构的文件存储的数据是按照关键字排序的顺序存储的。对于这样的文件，查找非常灵活，可以使用顺序扫描、折半查找等多种方式，修改比较简单。但是，数据的插入操作就非常耗时，平均需要移动一半的原来的数据。所以，这种文件常常不实时插入，而是将数据放入临时文件，在合适的时候合并插入，这样做的副作用是查找时可能需要查找临时文件。

链结构文件中的每个数据单位都附带一个指针，指向下一个数据单位。最后的数据单位的指针为结束符。这种文件的查找非常不方便，但是空间分配很灵活。其插入操作

非常简单，只要修改相应数据单位的指针即可。

块链文件是结合上面两种文件的优点而设计的，在块内使用向量结构，而块与块之间使用链式结构。其查找可以使用分块式查找，即先找到块，然后在块中查找。而插入操作由于只在块内移动数据，又比向量文件移动数据少，速度要快。插入操作的过程是：先找到需要把数据插入哪个块中，再查看块中是否有空。如果有，则需要移动块内的数据；如果没有，需要申请新的块。

Hash 又称散列，它提供一种机制，将关键字转换成地址，由于只是一个算法，所以速度很快。无论是查找、修改和插入，在理想情况下，都是一次到位。但 Hash 有冲突和数据溢出处理的问题。如果溢出处理不当，Hash 文件的性能会大为下降。

Hash 文件处理方法常常是，将存储区分为主数据区和溢出区，将主数据区分成若干个桶（bucket），每个桶的大小相等。要插入记录时，利用关键字直接计算它的存储地址，这个计算函数称为 Hash 函数。算法过程如下：

（1）将关键字转换为容易计算的数值。

（2）调整这个值的数量级，把它变换到适当的范围之内。

（3）紧缩范围，把调整后的数值精确地变换到桶号的范围之内。

（4）通常桶号不是地址，还需要查找“桶号-地址”表得到真正的存储地址。

由于 Hash 算法不是一一对应的算法，不同的关键字可能得到同样的地址，即使一个桶内可以放置多个数据单元，但也有满的可能性，需要考虑溢出的问题。开地址列的方法是不设置溢出区，将溢出的数据存储到下一个桶中。如果下一个桶也满了，则继续向下寻找空的桶。分离溢出区的方法是开辟溢出区，当有溢出数据时，就放入溢出区的桶中，设计时应该保证溢出区不会太大。分布式溢出区是将溢出区分布地放置在主数据区中，能够提高速度。

3.3.3　需要说明的问题

作为本节的结束，再说明两个问题。

1. 在数据交换中使用文件

当文件用于数据交换时，要注意的是，交换双方对文件格式的理解要一致。XML（eXtensible Markup Language，可扩展标记语言）已经在数据交换上被广泛采用，当机器的处理能力足够时，设计师可以优先考虑采用 XML 文件。有关 XML 的详细知识，请阅读本套丛书中的《系统分析师技术指南（2009 版）》的第 10 章。

2. 嵌入式文件系统

在嵌入式系统中，受限于成本和硬件设计，很多系统的数据存储似乎还没有采用文件方式，往往是将数据按照一定格式直接写进 Flash，这种格式完全由程序员掌握。而且，Flash 的操作必须是分块进行，实现起来也比较繁琐。现在已经有人对嵌入式的文件系统做了专门的研究，而且随着 Linux 等嵌入式操作系统慢慢成熟，嵌入式系统也越来越多

地采用文件作为数据存储方式。有关嵌入式系统设计的详细设计，请阅读第 9 章。

3.4 数据库的选择与设计

数据库管理系统经过了多年的发展，关系数据库产品已经如日中天，产品已经极大丰富，对于许多项目而言，大多数关系数据库产品都能满足从桌面系统到分布式系统的需要。

3.4.1 数据的组织

按照数据的组织，数据库可以分为网状数据库、层次数据库、关系数据库、支持面向对象的数据库和文档数据库。由于关系数据库具有数据结构化、最低冗余度、较高的程序与数据独立性、易于扩充、易于编制应用程序等优点，得到了广泛应用。而随着 Internet 的发展，以及人们对非结构化数据处理的要求不断提高，文档数据库开始了长足的发展。

1. 关系数据库

关系数据库的数学支持来源于关系代数，关系代数是一种抽象的查询语言，用对关系的运算来表达查询，作为研究关系数据语言的数学工具。关系代数的运算对象是关系，运算结果亦为关系。关系代数用到的运算符包括集合运算符、专门的关系运算符、算术比较符和逻辑运算符。SQL 语言是关系运算的一种简化版，几乎所有关系数据库都支持标准的 SQL 语句，而正是 SQL 的优势让关系数据库得到了广泛的应用。SQL 使全部用户（包括应用程序员、数据库管理员和终端用户）受益匪浅。SQL 的优势在于：

（1）非过程化语言。和过程化语言不同，不必告诉计算机怎么做，通过 SQL 只需要告诉计算机做什么，这种特性使用户更易集中精力于要得到的结果。

（2）统一的语言。SQL 可用于所有用户的 DB 活动模型，包括系统管理员、数据库管理员、应用程序员、决策支持系统人员，以及许多其他类型的终端用户。基本的 SQL 命令只需很少时间就能学会，最高级的命令在几天内便可掌握。

（3）所有关系数据库的公共语言。由于所有主要的关系数据库管理系统都支持 SQL 语言，用户可将使用 SQL 的技能从一个关系数据库系统转到另一个。所有用标准 SQL 编写的程序都是可以移植的。

2. 文档数据库

由于关系数据库关注于字符、数字等结构化数据的处理，而对其他非结构化的数据（如照片、声音、录像等）数据只是简单的存储和查看。而人们对这些数据的处理要求也慢慢从简单的存储、查看升级为识别、检索和加工。人们需要高效地存储、传播、分配和管理这类信息，需要一个文档数据库管理系统，其数据库的基本元素是文档。一个文档可以包含多种对象，例如，表格（可以是从某个关系数据库或电子表软件中得到的）、

格式化文本、Web 页面、图形、对象连接与嵌入（Object Linking and Embedding，OLE）对象、扫描的图像和传真件、声频或视频信号等多媒体信息。

3. 支持面向对象的数据库

面向对象数据库源于计算机编程语言中的面向对象技术。在关系数据库中，数据仅仅是数据，它不包含层次结构信息；而面向对象数据库可以将数据视为对象，数据是作为一个整体，包含了属性和方法，并能体现数据间的继承关系。

面向对象技术因为其技术的复杂性以及工业化成熟程度不够曾一度陷入困境，作为一种折衷，利用现有的优势，改造关系数据库并融入面向对象技术，即所谓的对象-关系数据库，则不失为上策。如今，IBM、Oracle 等知名厂商已经宣称其数据库产品支持面向对象技术。

3.4.2　数据的应用

按照应用来分，数据库可以分为桌面数据库、集中式网络数据库、分布式数据库、移动数据库和内存数据库。

1. 桌面数据库

桌面数据库主要运用于单机环境的系统，一般只接受本地计算机的访问请求，在功能、性能上也可能会略微逊色，在很多场合，设计师用它来取代文件作为本地存储。不过，有些优秀的产品在功能上和网络数据库已经别无二致，只是受到桌面计算机硬件性能限制，在性能上无法和网络数据库相比。

2. 集中式网络数据库

集中式网络数据库大多比较庞大，需要单独安装在服务器上，用来应对大量用户的繁重工作。这类数据库也是人们通常所说的“数据库”的同义词。网络数据库已经在各种系统中发挥了越来越重要的作用，人们在其性能、安全、备份、复制等都做了大量的工作。除了支持标准的 SQL，每个产品都会有不同程度的扩充。是否使用这些扩充，也是设计师需要谨慎对待的问题。如果使用扩充，可能会提高系统性能、减少实现的工作量，但系统的可移植性降低；如果不使用扩充，可能无法充分发挥某种产品的特别的优点，会增加工作量，延迟开发进度，但可以提高系统的可移植性。

3. 分布式数据库

分布式数据库是建立在计算机网络之上的一种数据库，而不是建立在单个计算机上的数据库。构成分布式数据库的数据存储在计算机网络的不同地点，由网络中计算机所运行的应用程序可访问不同地点的数据。

就其本质而言，分布式数据库系统的数据在逻辑上是统一的，而在物理上却是分散的。与集中式数据库相比，它有如下主要优点：

（1）当企业有众多分支机构，在地理上分布又广时，分布式数据库能有效地减少网络流量，同时数据库之间又保持必要的联系。

（2）当处理流量极大时，可采用分布式数据库进行负载均衡。

（3）和硬件采用冗余设计提供可靠性一样，这种软件的冗余也能使用增加投资的办法来提高整个系统的可靠性。

（4）当需要增加新的数据库实例使用时，对原有系统的影响最小。

分布式数据库系统虽然有诸多优点，但它同时也带来了许多新问题。例如，数据一致性问题、数据远程传递的实现、通信开销的增加等，这使得分布式数据库系统的开发变得较为复杂。同时，由于网络的发展，使得跨地区的广域网也有了相当的速度和可靠性，硬件的不断增强和降价，可以在不提高很多资金的情况下提供更高的处理性能。于是，一些原来采用分布式数据库处理的系统，现在却在向集中数据库发展。

有关分布式数据库的详细知识，请阅读本套丛书中的《系统分析师考试全程指导（2009 版）》的第 3.8 节。

4. 移动数据库

移动数据库是支持移动计算环境的分布式数据库。由于移动数据库系统通常应用在诸如掌上电脑、个人数字助理（Personal Digital Assistant，PDA）、车载设备、移动电话、收费终端等嵌入式设备中，因此，它又被称为嵌入式移动数据库系统。移动数据库和分布式数据库有一定的类似之处，都是把数据分散在不同的地方，不同之处在于分布式数据库之间往往存在健全的网络连接，而移动数据库往往是处于时断时续的状态。而且，分布式数据库一般不存在中央数据库，而移动数据库往往设计为和一个中央数据库相配合使用。

移动数据库最重要的问题在于移动数据库和中央数据库之间的数据一致性问题，设计师需要从以下方面来考虑：

（1）数据同步方式。设计师可以选择上载同步、下载同步和完全同步三种同步方式。

（2）冲突检测机制和冲突解决方案，冲突日志的记录。

（3）如何快速同步。系统同步时，只传递变化的数据，以节省同步时间。

（4）如何支持表的水平分割和垂直分割复制，降低移动数据库的大小。

（5）如何同异构数据源连接同步。

移动数据库由于其特性，还需要设计师考虑安全等问题。有关移动数据库的详细知识，请阅读本套丛书中的《系统分析师考试全程指导（2009 版）》的第 4.4 节。

5. 内存数据库

实时系统需要数据库有非常快的反应速度，而磁盘操作一直是数据库操作的瓶颈。虽然所有的数据库产品都会使用缓存技术来提高性能，但是，随着内存价格的不断下降，人们想到了数据常驻内存、事务的数据存取只涉及内存的内存数据库。

内存数据库的主要优点是由于摆脱了磁盘 I/O 处理的限制，取消了磁盘式数据库上很多额外的开销，事务的处理速度非常快。

内存数据库和普通数据库最大的不同除了数据在内存中，还有就是索引使用的不同

了。磁盘数据库系统典型的索引技术是 B-树索引。B-树结构的主要目的是减少完成数据文件的索引查找所需要的磁盘 I/O 数量，通过控制节点内部的索引值达到这个目的，在节点中包含尽可能多的索引条目（增加一次磁盘 I/O 可以访问的索引条目）。内存数据库广泛使用的 T-树是针对主存访问优化的索引技术。T-树是一种一个节点中包含多个索引条目的平衡二叉树，T-树的索引项无论是从大小还是算法上都比 B-树精简得多。T-树的搜索算法不分搜索的值在当前的节点还是在内存中的其他地方，每访问到一个新的索引节点，索引的范围就减少一半。

希赛教育专家提示：Hash 索引也是一种内存数据常用的索引方式。在前面已经了解到，Hash 算法是一种直接把关键字映射到地址的方法，使用 Hash 索引，可以只要通过一次 Hash 函数的计算，就可计算出数据存放的地址。

3.4.3　数据库设计实例

根据项目的实际情况选择合适的数据库，是对设计师的基本要求。有的设计师把关系数据库当作万能钥匙，无论什么样的系统都使用它。虽然关系数据库确实功能强劲，但它也不是万能的。

曾经有这样一个项目，客户需要交流大量创作性的文档，同时文档有许多模板，还希望能随时增加模板，这些文档甚至还需要许多人来完成其中不同的部分。不同的操作员有不同的权限，对文档的不同部分有不同的修改权，还有专门的评审小组对文档进行评审。需求分析完成之后，开发小组开始了工作，这个小组的设计师是经验丰富的关系数据库和 RAD（Rapid Application Development，快速应用开发）程序设计高手，所以他选用了“PowerBuilder+Sybase”的设计，为了达到客户需要，做了大量的工作。为了解决可以新增、修改模板的问题，就已经让设计师头痛了，还有随之而来的流程控制问题。小组工作了 6 个月后，在磕磕碰碰中完成了工作，可是客户对系统中抽象的概念显得非常困惑，繁琐的操作也让客户很难适应。

后来，不得不推翻原来的系统。新的设计师采用了“Lotus+Word”的设计，由于 Lotus 本身已经具备比较完善的文档管理和流程设计能力，小组得以轻松地在另外两个月以后重新开发系统。客户显然更满意于后来的系统，他们的主要工作是文档，对他们来说，有效完成工作是最重要的，而建立系统最重要的目的是为了简化实际工作中的审核部分。当然设计师不可能是上知天文，下知地理的天下奇才，但至少还是要知道自己所掌握的工具的优点和不足，在发现它们无法满足目标系统的工作时，需要启用新的工具。

1. 数据库设计

在根据系统的实际应用选择合适的数据库后，数据库的设计同样是设计师需要非常重视的问题。事实上有许多设计师已经把系统设计等同于数据库设计了，数据库设计涉及的内容众多，读者可以在书店轻松找到一打题目类似于“数据库设计”的如同砖头一样厚的书，这里无法涉及那么多方面，只能简单讨论一下数据库设计中最重要的问题。

对于大多数系统而言，数据库最重要的功能是数据存储的地方，而不是其他。因此，最重要的任务就在于如何将分析好的系统逻辑模型存储起来。由于面向对象设计方法和关系数据库的广泛采用，这个问题也变成了如何使用关系数据库存储对象。下面介绍如何将面向对象的设计映射到关系数据库中。

2. 如何将面向对象的设计进行关系数据库存储

当设计师完成了系统对象的设计，是开始考虑如何在关系数据库中存储自己所设计的对象的时候了。需要保存的对象也称为持久对象，显然，不是每个类都需要保存，那些以行为为主要特征的对象，例如，工厂类和容器类，在大多数情况下是没有保存的必要的。

一个普通的类可以映射为一个表或者多个表，当分解为多个表时，可以采用横切或者竖切的方法。

竖切常用于实例较少而属性众多的对象，一般是现实中的事物，把不同分类的属性映射成不同的表。把经常使用的属性放在主表中，而将其他一些次要的属性放在其他的表中。这样，能使设计师和实现者思维和操作起来方便一些，如果一个表有几百个字段，无论是设计时还是实现的时候，人的记忆都会显得不够的。

横切常用于记录与时间相关的对象，例如，成绩记录、缴费记录、设备的运行记录、检修记录等。由于经过或长或短的时间之后，这些对象就很少被查看，所以，往往在主表中只记录最近的对象，而在对应的历史表中记录全部对象。

类与类之间的关系比较复杂，如何存储这些关系也是设计师需要面对的问题。

3. 对继承关系的处理

（1）基类一个表，每个子类一个表。这是最保险和最常用的方法，理解起来比较容易。这个方法中继承关系的体现是基类对应的表中定义主键，而在子类对应的表中定义外键。

（2）每个子类一个表，没有基类表，把基类的所有属性都在每个子类表中复制一份。这样做性能可能会提高，但是，如果基类发生改变，所有的子类对应的表都需要改变。这种方法适用于子类的个数不多、基类属性比较少的情况。

（3）只有一个表，不但包括了基类的属性，而且包括了所有子类的所有属性，并增加一个字段区别子类的类型。这样，必然会存在空值字段。这种方法可以用于子类个数不多，而且大部分子类的属性相似的情况。

（4）当多重继承发生时，不但要使用标准的方法，而且要增加新的表来记录继承的情况。现在大多数语言不再支持多重继承，因此，除非设计师觉得确实有必要，否则，不必研究这种方法。

4. 对关系的处理

关联是一个类知道另一个类的属性和方法。普通的关联应该在数据库中保存为一个表。这个表有两个关键属性，分别是相关连的两个类的关键属性。例如，班级和学生两

个类，显然，在班级对象中会存在多个学生对象，这个关联就可以用一个“班级-学生”表来描述。

当多个类之间有关联时，设计师首先应该考虑是否使用三元关联的设计形式。如果确实有必要采用三元关联的方式，这个三元关联也要用一个表来描述，而这个表应该包含三个关键属性，分别代表三个关联对象的关键属性。

当然，如果是一对一的关联，这个表示关联的表可以退化到类映射的表中。

聚合关系的设计和关联类似，而合成关系和依赖关系更倾向于对象之间的行为关系，这种行为关系，关系数据库还没有什么好办法来保存。

3.5　网络环境下的系统设计

网络是人类继语言、文字、印刷术后的又一个伟大发明，如果说语言使得人与人的交流变得可能，文字使得未曾见面的人也能够交流，印刷术使得书本广泛传播，使得几乎所有人都能和作者交流，那么网络使得全世界的人都可以进行相互交流。而交流，正是人类进步的推动力之一。在这个网络时代，设计师无法回避网络这个现实。

谈到网络，人们首先想到的就是国际标准化组织（International Organization for Standards，ISO）的网络“七重天”，不过，在实践中严格按照这 7 个层次来设计的网络很少。随着 Internet 的发展，大多数人已经离不开它了，传输控制协议/互联网络协议（Transmission Control Protocol/Internet Protocol，TCP/IP）协议族已经成为事实上的计算机网络标准协议。虽然在不同的专业领域中，还存在大量的其他协议。不过，对于大多数设计师而言，了解 TCP/IP 协议族就可以满足普通环境下的设计了。

3.5.1　需要考虑的问题

如何实现不同网络及计算机间的互操作，成为计算机联网的最关键问题。经过 20 多年的研究，到 20 世纪 80 年代末 90 年代初，有了肯定的答案：就是采用 TCP/IP 协议。

TCP/IP 协议实际上就是在物理网上的一组完整的网络协议。用与开放系统互联（Open System Interconnection，OSI）同样的层次模型来描述 TCP/IP 网络协议组，则 TCP 是提供传输层服务，而 IP 是提供网络层服务。此外，由于 TCP/IP 是一组协议的代名词，所以它还包含许多别的协议。有关这些协议的详细知识，请阅读本套丛书中的《系统分析师考试全程指导（2009 版）》的第 5.2.3 节。

从总体上来说，系统设计师在进行网络程序设计时，需要考虑以下问题：

（1）通信方式和应用协议。设计师要根据当前项目的实际情况，选择合适的网络和网络协议。例如，使用用户数据报协议（User Datagram Protocol，UDP）发送重要的数据，那么设计师就不得不在重发机制上多费功夫了。当只在本系统之内通信，还可以考虑专门的协议；如果需要和其他系统通信，TCP/IP 协议显然是当前最好的选择。应用协

议是系统各站点之间的通信语言的语法，考虑应用协议时，应该考虑协议的无歧异、容易理解和实现。

（2）可靠性。网络天生就是不可靠的，无论网络设备商和系统集成商如何“吹捧”他们的网络如何健壮，作为系统设计师，都应该把网络设想为不可靠的，停电、设备故障、软件故障都可能产生网络中断。因此，在设计与网络相关的系统时，必须考虑网络不通和网络突然中断的情况。

（3）网络拥挤。人们把网络比喻成高速公路，可是高速公路都有塞车的时候。设计师需要考虑网络发生拥挤时应用系统的应对措施。典型的做法是使用缓存策略，这样，当网络拥挤时，系统能够等待一点时间，而用户在一定程度上不受到影响。

（4）安全性。安全性是一个无底洞，也就是说，如同在现实世界中“即使坐在墙根下看鸟，也可能被倒塌的墙砸着”一样，网络也没有绝对的安全。设计师在进行网络应用设计时，应该合理考虑安全性问题，主要是考虑安全问题发生的概率，以及问题发生时对系统的影响和损失有多大。在了解采用不同的安全级别花费的代价之后，才考虑采用什么级别的安全措施。有关安全性的详细知识，请阅读第 8 章。

希赛教育专家提示：系统安全性与系统性能也是一对矛盾体，各种安全措施的采取，都在某种程度上会损失系统的性能。

3.5.2 网络应用系统设计实例

现在，银行代收费业务似乎越来越红火，本节结合实际讨论代收费系统的设计。

显然，代收费系统使需要收费的公司（以下简称“公司”）免去了大量建设营业厅的投资，可以利用银行已经具备的网点资源，银行出租自己的资源来换取手续费，这是一个对双方都有好处的系统。要实现代收费基本有两种方式：批量代扣和实时缴费。批量代扣是公司的用户需要在银行开一个账户，到了固定的时间由银行根据公司提供的成本情况从账户上扣除相应的成本。实时缴费银行的营业厅必须能够实时知道公司用户的成本情况，并且能实时告诉公司其用户的缴费情况。这两种方式都被广泛采用，其中实时缴费对网络的要求更高。批量代扣方式可以使用其他文件交换方式，甚至双方可以不必联网。

出于安全考虑，双方只有一条物理的或逻辑的连接通道。在网络通道的两端，公司和银行各使用一台前置机完成通信工作。在公司的前置机安装两块网卡，一块和银行相连，一块和本地局域网相连。双方使用 TCP/IP 协议，在和银行相连的网卡上应该有防火墙控制，使其只接受固定计算机对本机固定端口的连接。这个防火墙可以是前置机上运行的软件，也可以是一个防火墙硬件产品。

前置机软件需要长期运行，监听双方协议好的端口，接收数据包，按照双方约定的数据格式分析数据包的内容。解释了银行的数据包中要求需要后，进行相应的处理，把处理结果返回给银行。这里实际上也是一个分布式的应用，如同在第 3.6 节描述的，这

是一个基于服务的分布式系统。在业务上包括费用查询、缴费、冲正、对账等操作。缴费成功后，还应该把发票内容给银行，银行打印收款凭证（发票）给用户，作为收费的凭证。

在这样的系统中，数据完整性和安全问题是设计的重点。由于网络的不可遇见的可能错误，会使数据包丢失。银行提交了收款信息却迟迟没有收到是否正确收款的回应，重发的过程中也同样可能发生丢包情况。系统设计师会发现，无法设计出一个完美的方案来，所以，几乎所有这样的系统最终都有一个对账的过程。每隔一段时间，通常是一天，银行都要把收费的记录副本给公司一份，并且公司以这个副本为真实收款数，对于那些公司认为收了款而银行认为没有收的，要更改收费的记录。在银行方面来说，真实的收款数必须和打印出的发票存根相匹配。

安全性在这样的系统中也格外重要。从网络硬件来说，双方只有一条通道相连，可以比较有效地防止对数据包的窃听和拦截。同时，防火墙（软件或者硬件）防止对系统非法端口的访问，能比较有效地防止系统留下的后门。

另外，系统的执行软件也需要采用一定的措施来保证安全问题：

（1）该软件只解释固定机器的数据。

（2）为了防止伪装，双方在数据通信之前需要进行握手：银行首先产生一个随机数，将这个随机数加密后发给公司；公司将这个加密的数据解密，同时也产生一个随机数，将两个数合并，再次加密后回复给银行。银行将这两个数解密后，将前一个数和自己发送的数据相比较。比较相符后将后面的数加密后传给公司，公司将其解密，和刚才发送的数据进行比较。如果相符，则握手过程结束，可以进行通信了。双方的密钥不同，并定期更换，更换密钥则使用其他途径进行。

（3）握手过程结束后的业务通信数据也可以加密。这个密钥和握手的应该不同。

如果实时缴费业务量很大，公司这一端的一台前置机性能无法满足要求时，可以使用负载均衡技术来把任务分给多台机器完成。这样，和银行连接的机器上只做一个负载均衡的软件，而多个前置机的软件基本可以不变。

3.6　分布式系统设计

网络极大地扩展了计算机的应用范围，同时，由于升级到更强的服务器的费用常常远高于购买多台档次稍低的机器，更何况虽然计算机有了长足的发展，可是单台计算机的功能仍然十分有限，利用联网的计算机协同工作，共同完成复杂的工作成为比较而言相对成本较低的选择，而且，还可以完成单台计算机所无法完成的任务。分布式系统使得这一目标成为可能。另外一方面是，网络本质上并不可靠，特别是远程通信，分布式系统还带来了并发和同步的问题。

1．基于实例的协作

在这种方式中，所有的实例都处理自己范围内的数据，这些对象实例的地位是相同的，当一个对象实例必须要处理不属于它自己范围的数据时，它必须和数据归宿的对象实例通信，请求另外一个对象实例进行处理。请求对象实例可以启动对象、调用远程对象的方法，以及停止运行远程实例。

基于实例的协作具有良好的灵活性，但由于实例之间的紧密联系复杂的交互模型，使得开发成本提高，而且，由于实例必须能够通过网络找到，所以通信协议必须包括实例的生存周期管理，这使得基于实例的协作大多只限于统一的网络，对于复杂的跨平台的系统就难以应付。所以基于实例的协作适用于比较小范围内网络情况良好的环境中，这种环境常常被称为近连接。这种情况下对象的生存周期管理所带来不寻常的网络流量是可以容忍的。

使用基于实例的协作常常使用被称为“代理”的方法，某个对象实例需要调用远程对象时，它可以只和代理打交道，由代理完成和远程对象实例的通信工作：创建远程对象，提交请求、得到结果，然后把结果提交给调用的对象实例。这样，这个对象实例甚至可以不知道自己使用了远程对象。当远程对象被替换掉（升级）时，对本地代码也没有什么需要修改的地方。

2．基于服务的协作

这种方法试图解决基于实例的协作的困难。它只提供远程对象的接口，用户可以调用这些方法，却无法远程创建和销毁远程对象实例。这样就减少了交互，简化了编程，而且使得跨平台协作成为可能。同样由于只提供接口，这种协作方式使得对象间的会话状态难以确定，而且通信的数据类型也将有所限制，基本上很难使用自定义的类型。基于服务的协作适用于跨平台的网络，网络响应较慢的情况，这种环境常称为远连接，这时，简化交互性更为重要，而频繁的网络交换数据会带来难以容忍的延时。

基于服务的协作通常采用层次式体系结构，高层的应用依赖于低层的对象，而低层对象实例的实现细节则没有必要暴露给高层对象，这种安排使得高层的实现不受低层如何实现的影响，同时，当低层服务修改时，高层的服务也不应该受到影响。

系统设计师在进行设计时，通常会倾向于比较细致的设计，对象往往提供了大量的操作和方法，响应许多不同的消息，以增加达到系统的灵活性、可维护性等。这在单个系统中没有什么问题，当考虑分布式系统的设计时，这种细致的设计所带来对对象方法的大量调用会比较严重的影响性能，所以在分布式系统中，倾向于使用大粒度的设计方式，往往在一个方法中包含了许多参数，每个方法基本上代表了一个独立的功能。当然这样的设计使得参数的传递变得复杂，当需要修改参数时，需要对比较大范围的一段过程代码进行修改，而不是像小粒度设计一样，只需要修改少量的代码。

本套丛书中的《系统分析师考试全程指导（2009 版）》的第 13.5.2 节给出了分布式系统设计的一个实例，有兴趣的读者可以参考。

3.7　运行环境的集成与设计

在设计一个新系统时，系统设计师必须考虑目标系统的运行环境问题，人们往往认为软件应该能够在任何环境中运行，但事实并非如此。软件的运行环境是指系统运行的设备、操作系统和网络配置。本节给出软件运行的几个典型环境，系统设计师可以从这几种典型环境中选择适合自己的目标系统的环境，也可以将这些典型环境做一些组合，来满足自己设计的系统的特殊要求。

1. 集中式系统

早期的计算机系统没有什么可以选择的，除了集中式系统。所有的操作都集中于一台主机中，而操作员必须在主机的附近操作，结果也在附近给出。目前，这种系统仍然广泛应用于批处理应用系统中。

集中式系统常见于银行、保险、证券行业，它们含有大规模的处理应用。而现在流行的电子商务又给了大型处理机注入了新的活力，人们发现电子商务要面对大量的事务，需要大型处理机来处理。但是，实践中很少单独使用集中式系统，因为大量的系统需要处理在地理上分布得很远的连接请求，这些请求有的需要实时响应，并可能要发送到其他某个地方的一个集中式系统。所以，在现代的系统中，集中式系统通常是某个分布式系统的一个环节。

集中式系统的组成结构如下。

（1）单计算机结构：这种结构简单、容易维护，但是处理能力受到限制。

（2）集群结构：由多台计算机组成，这些计算机具有类似的硬件平台、操作系统等。通常采用负载均衡、资源共享等方式实现更大的处理能力和容量。

（3）多计算机结构：由多台计算机组成，这些计算机之间操作环境可能不同。适用于当系统可以分解成多个不同的子系统时。

2. 分布式系统

由于网络的普遍延伸，分布式系统越来越成为大型系统的首选环境。分布式系统必须基于网络，这个网络可以是在一个地域内的局域网，也可以是跨越不同城市乃至国家的广域网。与集中式的计算机环境相比，分布式系统有着多种多样的形式。这也给设计师在确定系统运行环境时带来一定的烦恼。

在传统的 C/S 中，系统由提供服务的服务器和发起请求、接收结果的客户端构成。这种结构是一种可以使用很多方式实现的通用结构模型。并非只限于数据库的 C/S 结构，典型的还有网络打印服务系统，现在流行的网络游戏也是基于这种结构的。

多层结构是 C/S 结构的扩展，典型的分为存储数据的数据库服务器作为数据层，实现商业规则的程序作为逻辑层，管理用户输入输出的表示层所组成的三层结构。当系统更复杂时，可以再增加其他层次构成多层结构。

多层结构形式复杂，功能多样。实现多层结构常常需要实现不同层次间通信的专用程序（中间件），中间件大多数实现远程程序调用、对象请求调度等功能。

现代企业级的计算机系统大量的基于分布式结构。支持分布式系统的软件也曾经如同雨后春笋。系统如何分层、如何处理分布带来的同步等问题也就同样在考验设计师。

3. Internet、Intranet 和 Extranet

Internet 是全球的网络集合，使用通用的 TCP/IP 协议来相互连接，提供电子邮件、文件传输、远程登录等服务。Intranet 是私有网络，只限于内部使用，也使用 TCP/IP 协议。Extranet 是一个扩展的 Intranet，它包括企业之外的和企业密切相关合作的其他企业。Extranet 允许分离的组织交换信息并进行合作，这样就形成了一个虚拟组织，例如，虚拟专用网络（Virtual Private Network，VPN）技术允许在公用网络上构架只对组织内部开放的服务。

Web 同样基于 C/S 结构（或者说是 C/S 结构的一个变种），实际上 Web 接口是一个通用的接口，不是只能使用浏览器的协议，它同样能够在普通的程序中使用。Internet 和 Web 已经给系统设计师提供了一个非常富有吸引力的选择方案。它的优势在于：它们已经成为网络的事实标准，支持它们的软件已经广泛存在于全世界的计算机中，而且通信费用已经下降到很有竞争力的水平。从某种程度上来说，企业可以把 Internet 当作自己廉价的广域网。

当然，事物总有相反的一面，当系统设计师试图采用 Internet 时，必须考虑其不利的一面。Internet 的安全性过去是，现在是，以后仍然是设计师头痛的问题。其他诸如可靠性、系统吞吐量、不断发展的技术和标准都是影响设计师选择它们作为运行环境的不利因素。

系统设计师应该根据目标系统的实际需要来选择不同的运行环境。不过，已经有越来越多的公司提供支持 Internet 和电子商务的接口支持。

本章参考文献

[1] 孔璐. 数据库原理与开发应用技术. 北京：国防工业出版社，2004

[2] 梅宏. 软件工程——实践者的研究方法. 第 5 版. 北京：机械工业出版社，2002

[3] 吴丹，史争印，唐忆. 软件工程理论与实践. 第 2 版. 北京：清华大学出版社，2003

[4] 朱群雄，李芳，汪晓男. 系统分析与设计. 第 2 版. 北京：电子工业出版社，2003

[5] 宛延闿，定海. 面向对象分析和设计. 北京：清华大学出版社，2001

第4章　软 件 设 计

从功能上的划分来看，软件设计应该是软件设计师的工作，但作为一名能够指导软件师工作的系统分析师，首先自己必须懂得软件设计的基本原则和理论，掌握基本的软件设计方法，具有丰富的软件设计经验。

4.1　结构化设计

结构化设计（Structured Design，SD）方法是 SA 方法的延续，它是基于模块化、自顶向下逐层细化、结构化程序设计等技术基础上发展起来的。该方法实施的过程如下：

（1）总结出系统应有的功能，对一个功能，从功能完成的过程考虑，将各个过程列出，标识出过程转向和传递的数据。这样，可以将所有的过程都画出来。

（2）细化数据流。确定应该记录的数据。

（3）分析各过程之间的耦合关系，合理地进行模块划分以提高它们之间的内聚性。

4.1.1　设计基本原则

在一节不和谐的课堂里，老师叹气道："要是坐在后排聊天的同学能像中间打牌的同学那么安静，就不会影响到前排睡觉的同学"。这个故事告诉我们，如果不想让坏事传播开来，就应该把坏事隐藏起来，"家丑不可外扬" 就是这个道理。为了尽量避免某个模块的行为去干扰同一个系统中的其他模块，在设计模块时就要注意信息隐藏。应该让模块仅仅公开必须要让外界知道的内容，而隐藏其他一切内容。

1．信息隐蔽

在软件设计中，同样有信息隐蔽原则。Parnas 提出：在概要设计时列出将来可能发生变化的因素，并在模块划分时将这些因素放到个别模块的内部。也就是说，每个模块的实现细节对于其他模块来说是隐蔽的，模块中所包含的信息（包括数据和过程）不允许其他不需要这些信息的模块使用。这样，在将来由于这些因素变化而需修改软件时，只需修改这些个别的模块，其他模块不受影响。

信息隐蔽技术不仅提高了软件的可维护性，而且也避免了错误的蔓延，改善了软件的可靠性。现在，信息隐蔽原则已成为软件工程学中的一条重要原则。

希赛教育专家提示：信息隐蔽原则并不是结构化设计方法所独有的，在面向对象设计技术中，信息隐蔽也是一个很重要的原则，主要体现在类和对象的封装性上。

2．模块独立

模块独立性是指软件系统中每个模块只涉及软件要求的具体子功能，而和软件系统中其他的模块接口是简单的。模块独立的概念是模块化、抽象、信息隐蔽和局部化概念的直接结果。

一般采用两个准则度量模块独立性，分别是模块间耦合和模块内聚。耦合是模块之间的相对独立性（互相联系的紧密程度）的度量。模块之间的联系越紧密，联系越多，耦合性就越高，而其模块独立性就越弱；内聚是模块功能强度（一个模块内部各个元素彼此结合的紧密程度）的度量。一个模块内部各个元素之间的联系越紧密，则它的内聚性就越高，相对地，它与其他模块之间的耦合性就会减低，而模块独立性就越强。因此，模块独立性比较强的模块应是"高内聚、低耦合"的模块。

模块的内聚类型通常可以分为7种，根据内聚度从高到低的排序如表4-1所示。

表4-1 模块的内聚类型

内 聚 类 型	描 述
功能内聚	完成一个单一功能，各个部分协同工作，缺一不可
顺序内聚	处理元素相关，而且必须顺序执行
通信内聚	所有处理元素集中在一个数据结构的区域上
过程内聚	处理元素相关，而且必须按特定的次序执行
瞬时内聚	所包含的任务必须在同一时间间隔内执行（如初始化模块）
逻辑内聚	完成逻辑上相关的一组任务
偶然内聚	完成一组没有关系或松散关系的任务

模块的耦合类型通常也分为7种，根据耦合度从低到高排序如表4-2所示。

表4-2 模块的耦合类型

耦 合 类 型	描 述
非直接耦合	没有直接联系，互相不依赖对方
数据耦合	借助参数表传递简单数据
标记耦合	一个数据结构的一部分借助于模块接口被传递
控制耦合	模块间传递的信息中包含用于控制模块内部逻辑的信息
外部耦合	与软件以外的环境有关
公共耦合	多个模块引用同一个全局数据区
内容耦合	一个模块访问另一个模块的内部数据；一个模块不通过正常入口转到另一模块的内部；两个模块有一部分程序代码重叠；一个模块有多个入口

除了满足以上两大基本原则之外，通常在模块分解时还需要注意：保持模块的大小适中；尽可能减少调用的深度；直接调用该模块的次数应该尽量多，但调用其他模块的次数则不宜过多；保证模块是单入口、单出口的；模块的作用域应该在模块之内；功能

应该是可预测的。

4.2.2　模块结构

模块就如同人的器官，具有特定的功能。人体中最出色的模块设计之一是手，手只有几种动作，却能做无限多的事情。人体中最糟糕的模块设计之一是嘴巴，嘴巴将最有价值但毫无相干的几种功能如吃饭、说话、亲吻混为一体，使之无法并行处理，真乃人类之不幸。

结构图（Structured Charts，SC）是准确表达程序结构的图形表示方法，它能清楚地反映出程序中各模块间的层次关系和联系。与 DFD 反映数据流的情况不同，SC 反映的是程序中控制流的情况。例如，图 4-1 为某大学教务管理系统的结构图。

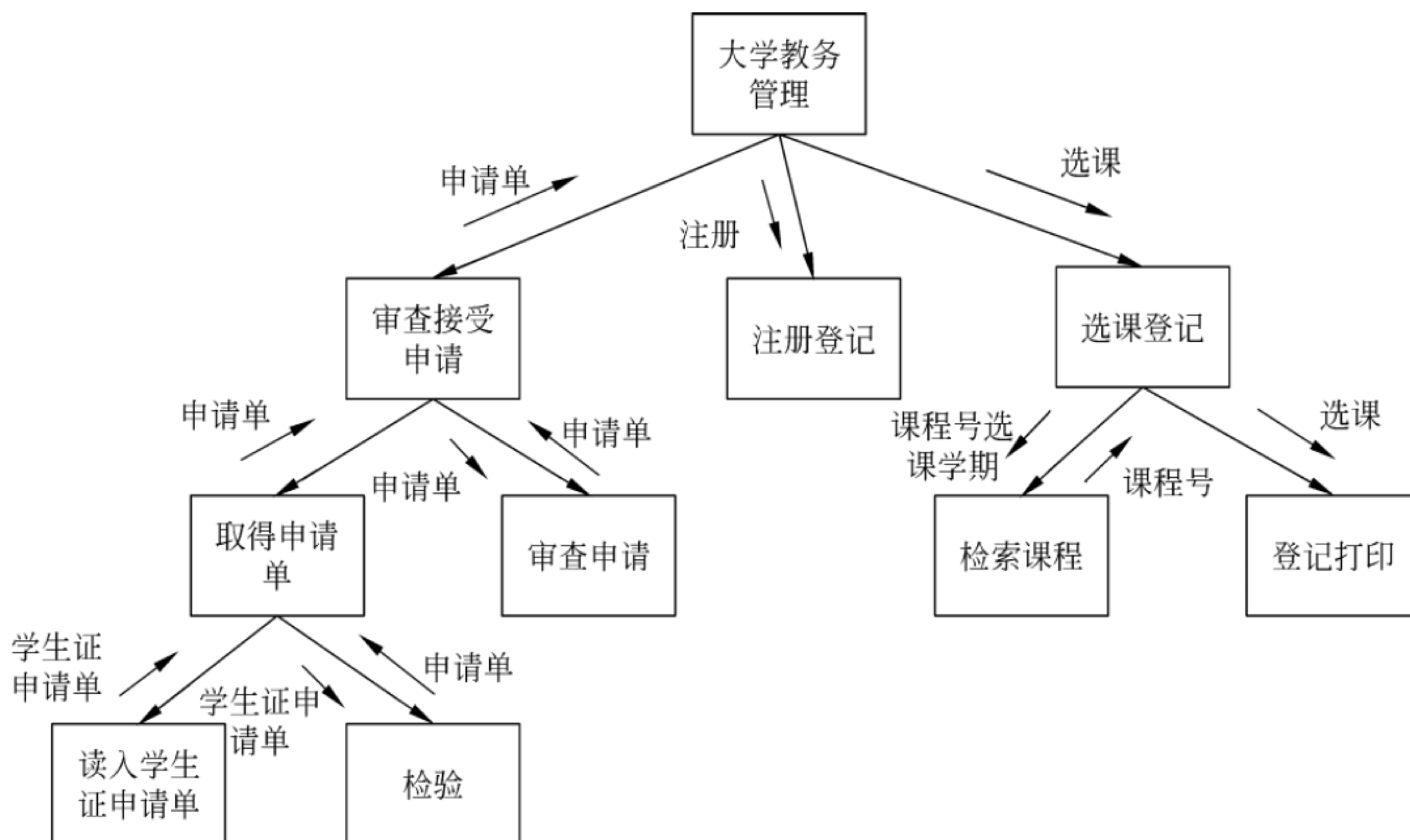


图 4-1　大学教务管理系统结构图

在 SC 中，主要成分有模块及调用关系，以及一些辅助控制符号。

（1）模块。以矩形框表示，框中标有模块的名字。对于已定义（或者已开发）的模块，则可以用双纵边矩形框表示，如图 4-2 所示。

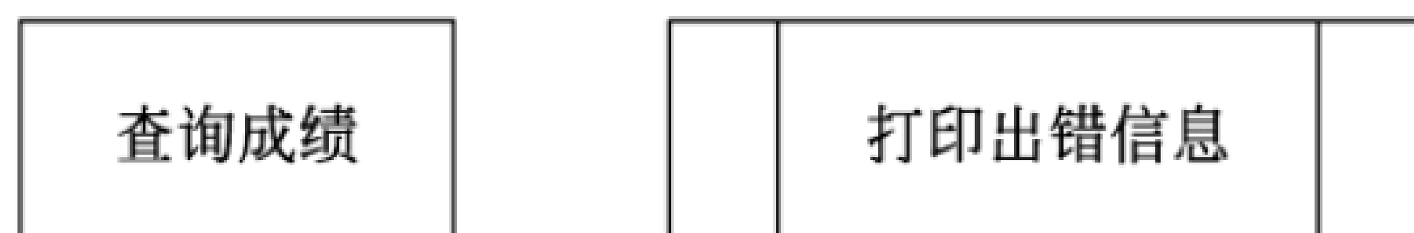


图 4-2　模块的表示

（2）模块间的调用关系。两个模块，一上一下，以箭头相联，上面的模块是调用模

块，箭头指向的模块是被调用模块。例如，在图 4-3 中，模块 A 调用模块 B。在一般情况下，如果不会引起混淆，箭头表示的联线可以用直线代替。

（3）模块间的通信。以表示调用关系的长箭头旁边的短箭头表示，短箭头的方向和名字分别表示调用模块和被调用模块之间信息的传递方向和内容。例如，在图 4-3 中，首先，模块 A 将信息 C 传给模块 B，经模块 B 加工处理后的信息 D 再传回给 A。

（4）辅助控制符号。当模块 A 有条件地调用模块 B 时，在箭头的起点标以菱形。模块 A 反复地调用模块 D 时，另加一环状箭头，如图 4-4 所示。在 SC 中，条件调用所依赖的条件和循环调用的循环控制条件通常都无需注明。

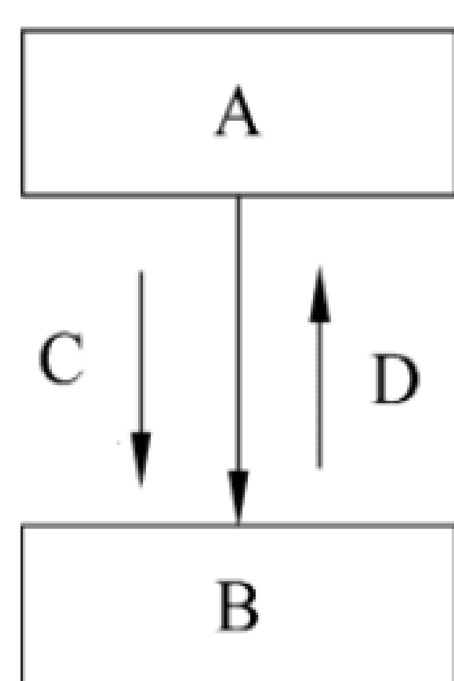


图 4-3　模块的调用关系及信息传递关系的表示

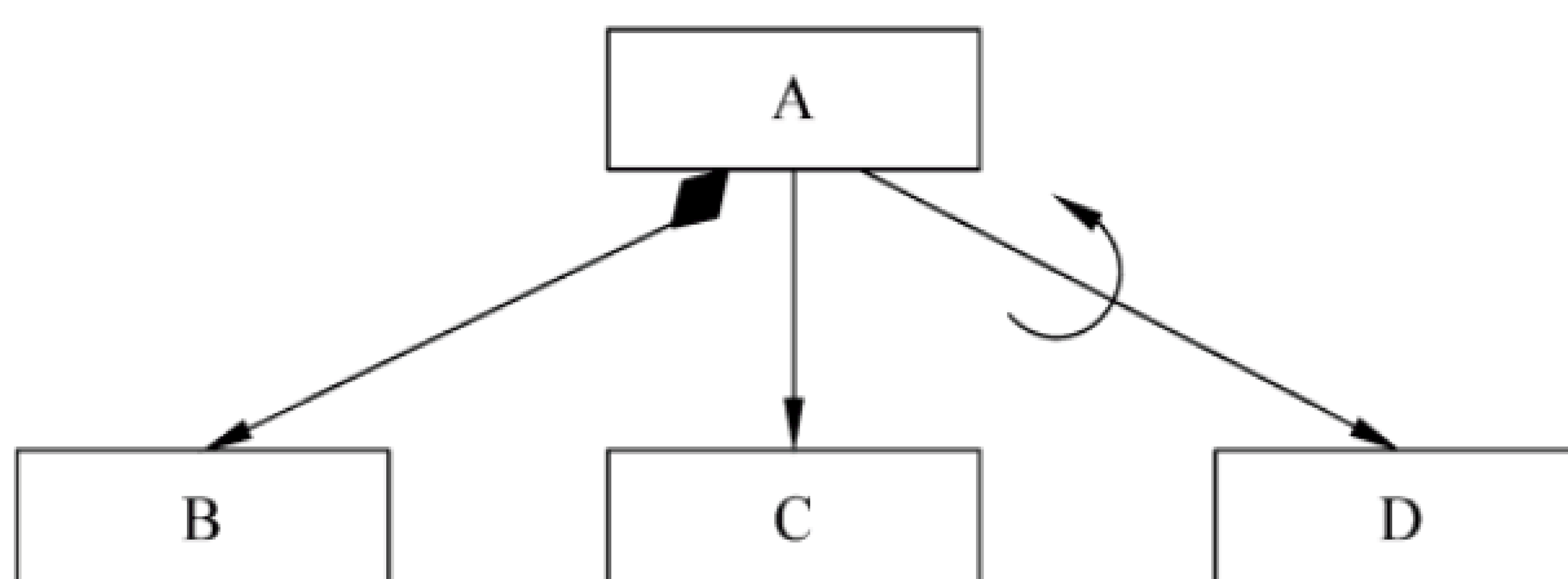


图 4-4　条件调用和循环调用的表示

在 SC 中，不能再分解的底层模块称为原子模块。如果一个软件系统的全部实际加工都由原子模块来完成，而其他所有非原子模块仅仅执行控制或协调功能，这样的系统就是完全因子分解的系统。如果系统结构图是完全因子分解的，就是最好的系统。但实际上，这只是力图达到的目标，大多数系统做不到完全因子分解。

一般来说，SC 中可能出现图 4-5 所示的 4 种类型的模块。

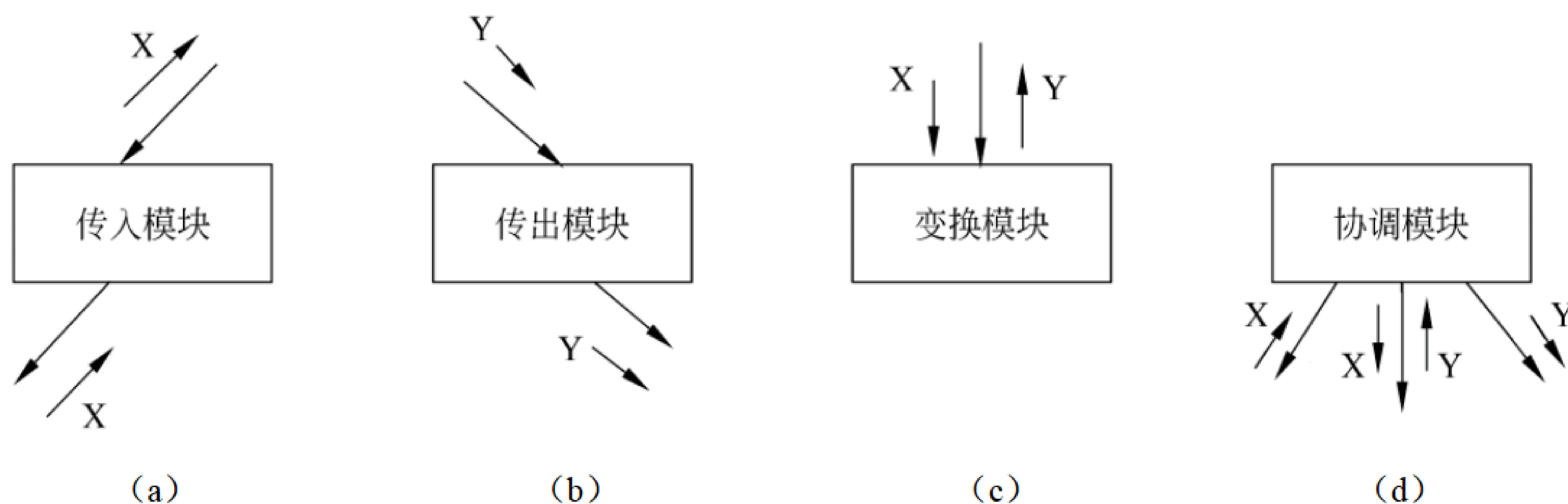


图 4-5　4 种模块类型

（1）传入模块：从下属模块取得数据，经过某些处理，再将其传送给上级模块。它传送的数据流叫做逻辑输入数据流。

（2）传出模块：从上级模块取得数据，进行某些处理，传送给下属模块。它传送的数据流叫做逻辑输出数据流。

（3）变换模块：从上级模块取来数据，进行特定处理后，送回原上级模块。它加工

的数据流叫做变换数据流。

（4）协调模块：对其下属模块进行控制和管理的模块。在一个好的系统结构图中，协调模块应在较高层出现。

希赛教育专家提示：系统结构图着重反映的是模块间的隶属关系，即模块间的调用关系和层次关系。它和程序流程图（常称为程序框图）有着本质的差别。程序流程图着重表达的是程序执行的顺序以及执行顺序所依赖的条件。系统结构图则着眼于软件系统的总体结构，它并不涉及模块内部的细节，只考虑模块的作用，以及它和上、下级模块的关系。而程序流程图则用来表达执行程序的具体算法。

4.2.3　常用的系统结构图

在结构化分析和设计技术中，通常存在着两种典型的问题类型，变换型问题和事务型问题，它们的 DFD 和系统结构图都有明显的特征。

1．变换型系统结构图

在数据处理问题中，通常会遇到这样一类问题，即从程序“外部”取得数据（例如，从键盘、磁盘文件等），对取得的数据进行某种变换。然后，再将变换得到的数据传回给“外部”。其中取得数据的过程称为传入数据流程，变换数据的过程称为变换数据流程，传回数据的过程称为传出信息数据流程，如图 4-6 所示。

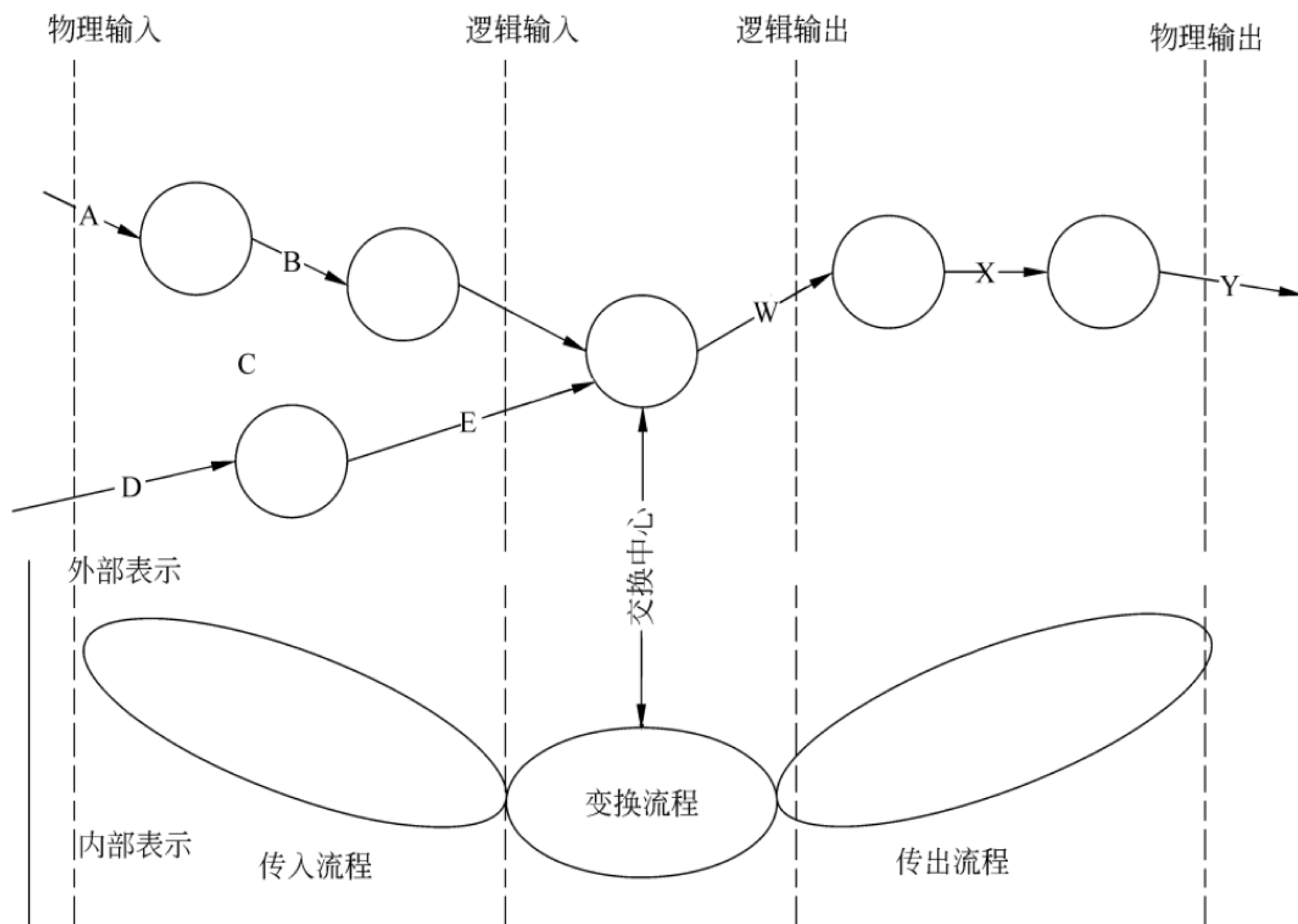


图 4-6　变换型问题

当DFD或其中某一段数据流表现出上述特征时，该DFD表示的就是一个变换型问题。完成数据变换的加工称为变换中心。变换型问题 DFD 基本形态及其对应的基本结构图分别如图4-7（a）和（b）所示。

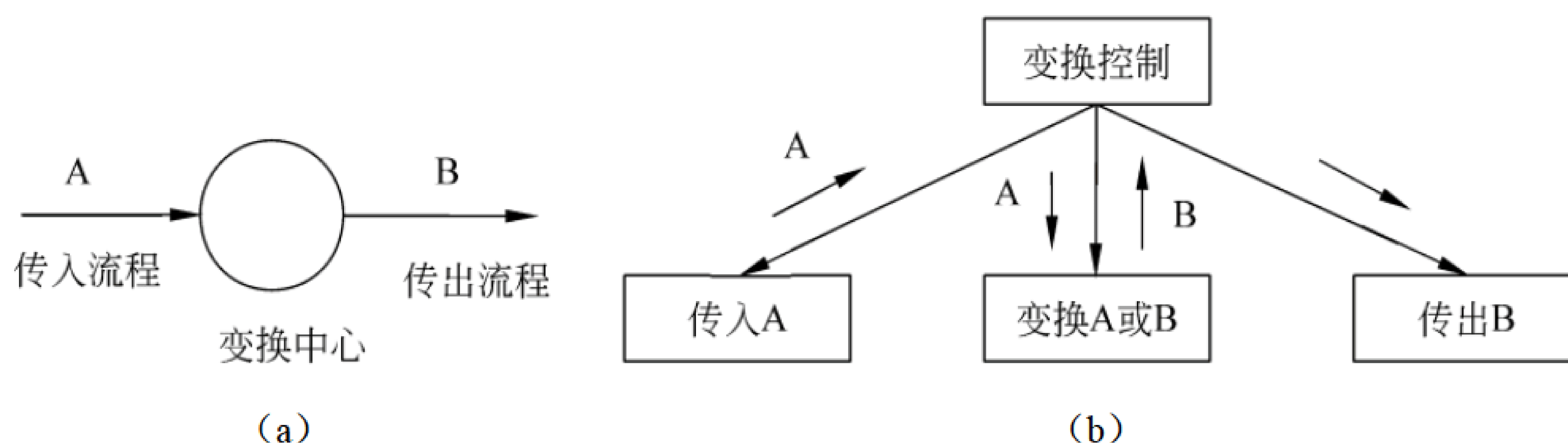


图4-7　基本变换型问题DFD及其结构图

按照图4-7表示的基本映射关系，可以得到图4-6对应的SC，如图4-8所示。

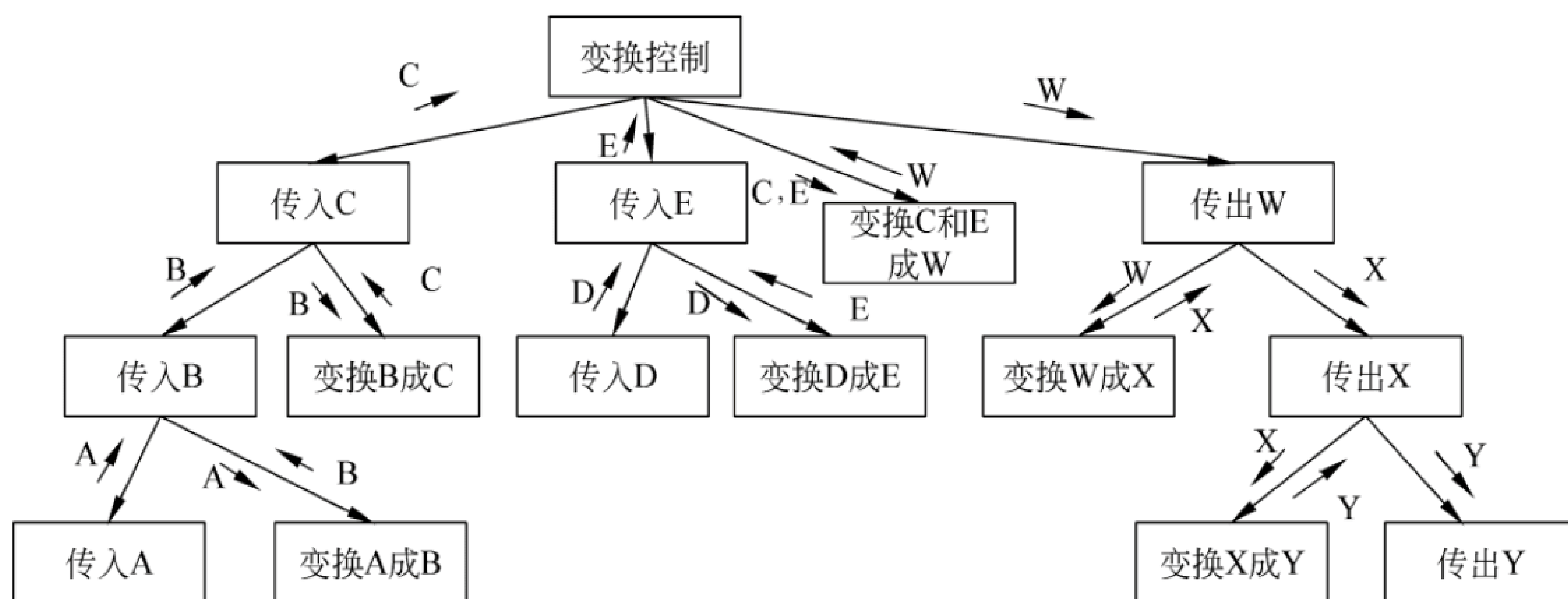


图4-8　变换型问题结构图

变换控制模块首先获得控制，然后控制沿着结构到达底层的“传入A”模块，物理输入数据A由“传入A”模块读入后，从底层逐步向上传送。在传送过程中，数据经“变换A成B”、“变换B成C”等模块的预处理，逐渐变换成纯粹的逻辑输入C。接着，在变换控制模块的控制下，将逻辑输入经变换中心模块“变换C和E成W”处理后，变换成逻辑输出W，再从顶层逐步向下传送。在这一传送过程中，数据经“变换W成X”、“变换X成Y”等模块的后处理，逐渐变换成适当的输出形式，最后由“传出Y”模块完成物理输出。

2．事务型系统结构图

在实际应用中，还常常会遇到另一类问题，即通常在接受某一项事务后，根据事务的特点和性质，选择分派给一个适当的处理单元，然后给出结果，如图4-9所示。

这类问题就是事务型问题。它的特点是，数据沿着接收分支把外部数据转换成一个事务项，然后计算该事务项的值，并根据它的值从多条数据流中选择其中的某一条数据流。发出多条数据流的处理单元称为事务中心。这类问题的典型结构如图4-10所示。事

务控制模块按所取得事务的类型，选择调用某一个处理事务模块。

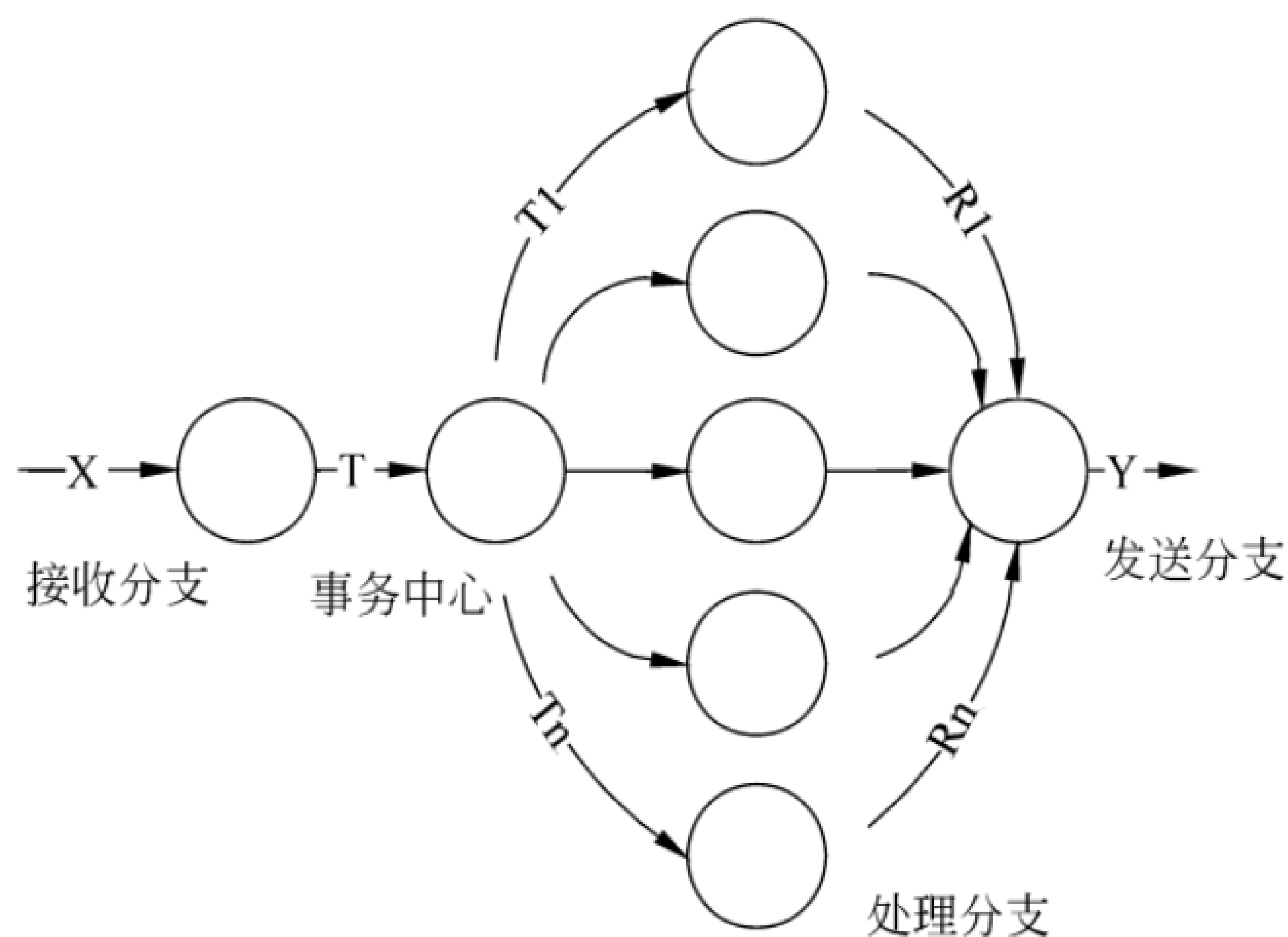


图 4-9　事务型问题

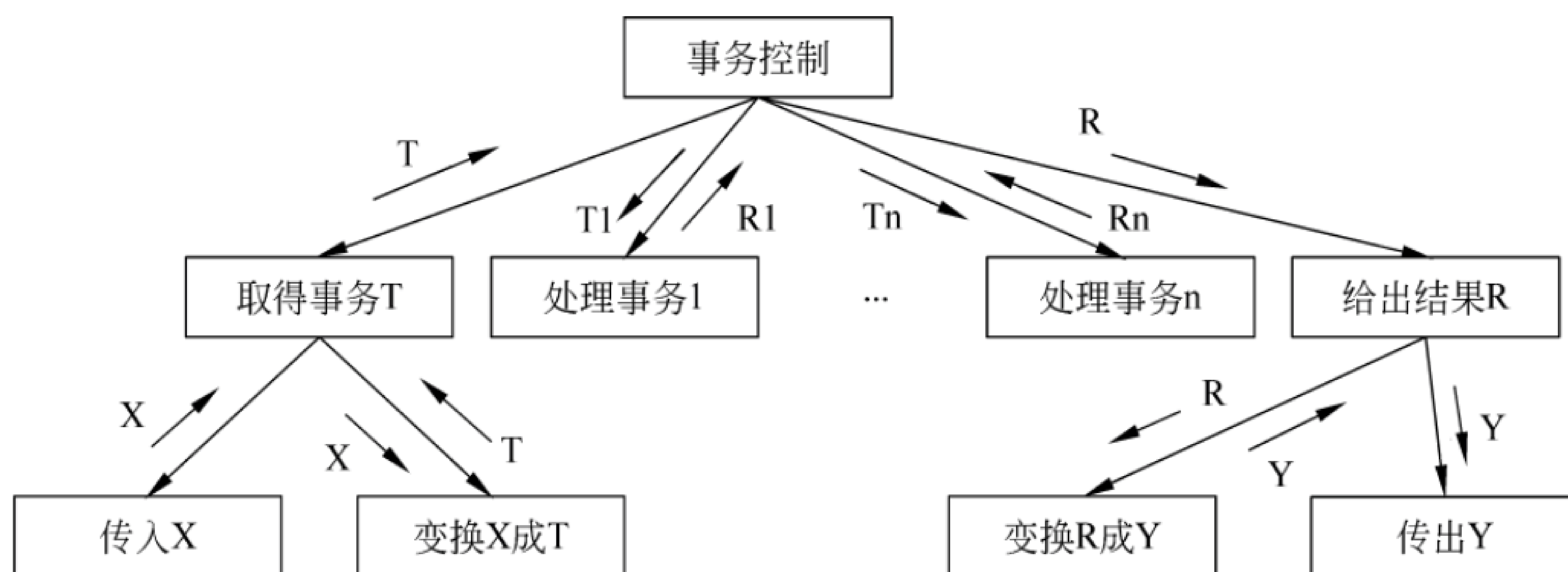


图 4-10　事务型问题结构图

3．混合型系统结构图

一些大型问题既不是单纯的变换型问题，也不是单纯的事务型问题，而是混合在一起的混合型问题。图 4-11 表示的是一个混合型问题 DFD，图 4-12 是图 4-11 相对应的 SC。

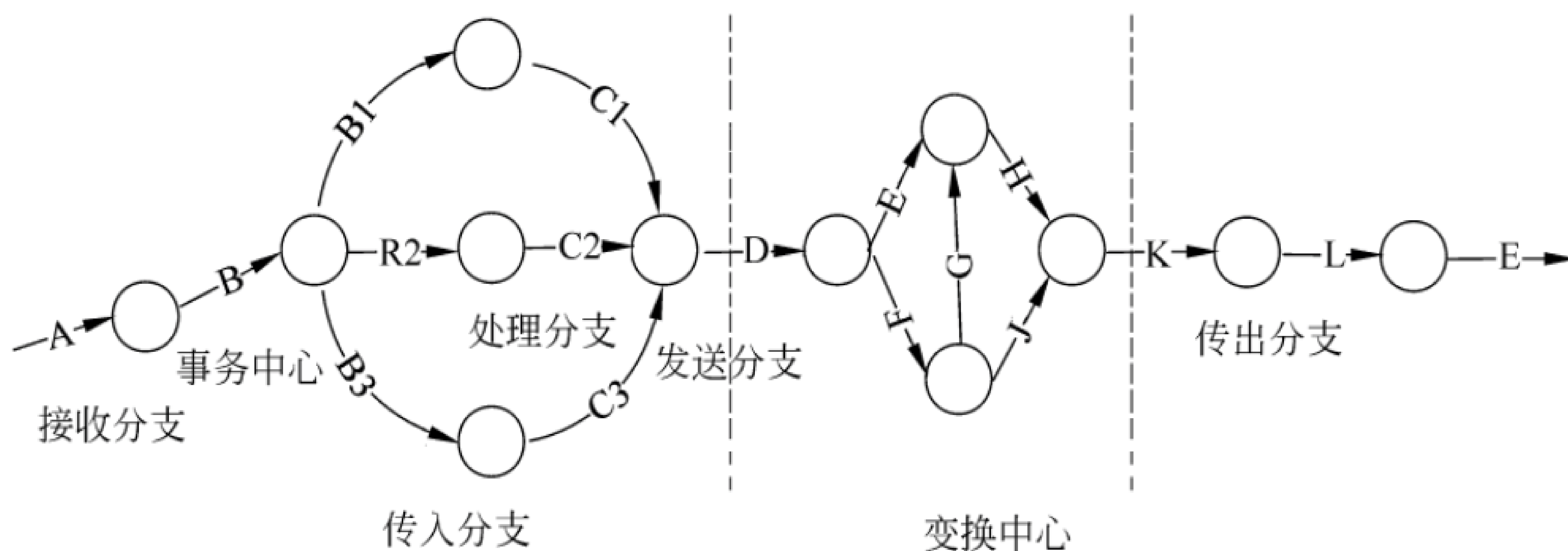


图 4-11　混合型问题 DFD

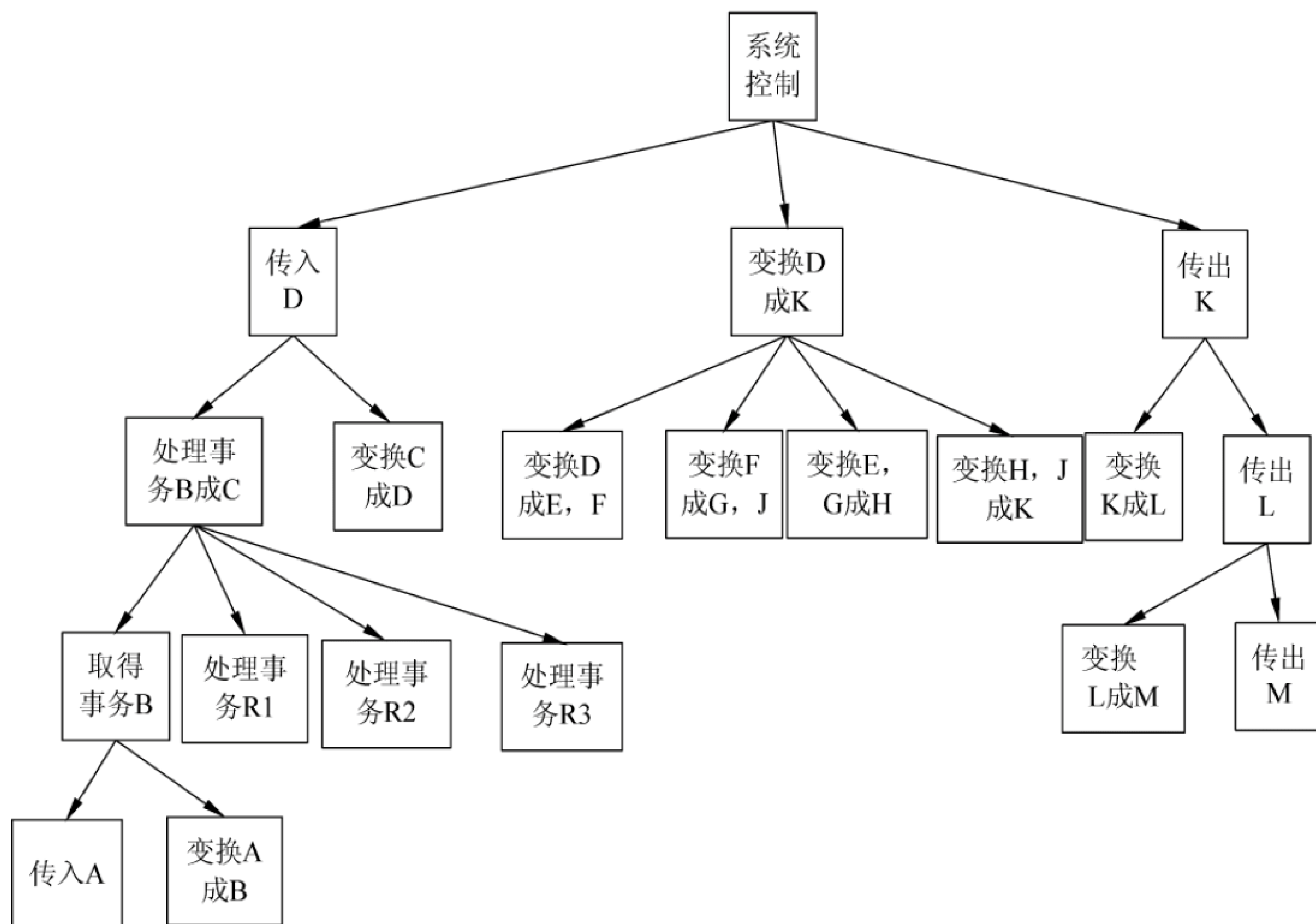


图 4-12　混合型问题 SC

对于这种混合型问题，一般以变换型问题为主。首先，找出变换中心，设计出结构图的上层；然后，根据 DFD 的各部分具体类型分别映射得到它们的 SC。例如，对于图 4-12 所表示混合型问题，从整体上可以将其看成是一个从 A 到 M 的变换型问题，从 D 到 K 之间的变换是变换中心；从 A 到 D 是传入分支，具有事务型问题的特点；从 K 到 M 是传出分支。因此，该混合型问题 SC 的上层可以由“传入 D”模块、“变换 D 成 K”模块和“传出 K”模块组成，“传入 D”模块的下层结构图由从传入分支映射得到的事务型问题结构图组成，“变换 D 成 K”模块和“传出 K”模块的下层结构图可以按通常的变换型问题映射方法获得。

4.3　面向对象设计

分析阶段主要是明确用户的需求，设计阶段则主要是确定实现用户需求的方法，即怎样做才能满足用户需求，并构造出系统的实现蓝图。面向对象设计（Object-Oriented Design，OOD）也是如此，只不过是引入了一些面向对象的概念和原则，用以指导设计工作。OOD 首先从 OOA 的结果开始，并将其从问题域映射到实现域。为满足实现的需要，还要增加一些类、结构及属性和服务，并对原有类及属性进行调整。此外，还要完成应用控制、人机交互界面的设计等。

4.3.1　Booch 方法

在 OOD 中，Booch 方法强调基于类和对象的系统逻辑视图与基于模块和进程的系统物理视图之间的区别。他还区别了系统的静态模型和动态模型。然而，Booch 方法偏向于系统的静态描述，对动态描述支持较少。

动态逻辑模型描述对象之间的“互相作用”，互相作用通过一组协同的对象、对象之间消息的有序序列、参与对象的可见性定义，来描述系统运行时的行为。Booch 方法用对象交互作用图描述重要的互相作用，显示参与的对象、对象之间按时间排序的消息；用可见性图描述互相作用中对象的可见性。对象的可见性定义了一个对象如何处于向它发送消息的方法的作用域之中。例如，它可以是方法的参数、局部变量、新的对象，或当前执行方法的对象的一部分。

静态物理模型通过模块描述代码的布局，描述软件的进程和线程体系结构。

Booch 方法的过程包括以下步骤：

（1）在给定的抽象层次上识别类和对象。包括找出问题空间中关键的抽象和产生动态行为的重要机制。开发人员可以通过研究问题域的术语发现关键的抽象。

（2）识别对象和类的语义。建立前一阶段识别出的类和对象的含义。开发人员确定类的行为（即方法）和类及对象之间的互相作用（即行为的规范描述）。该阶段利用状态转移图描述对象的状态的模型，利用时态图（系统中的时态约束）和对象图（对象之间的互相作用）描述行为模型。

（3）识别类和对象之间的关系。描述静态和动态关系模型。这些关系包括使用、实例化、继承、关联和聚集等。类和对象之间的可见性也在此时确定。

（4）实现类和对象。考虑如何用选定的编程语言实现，如何将类和对象组织成模块。

希赛教育专家提示：这 4 种活动不仅仅是一个简单的步骤序列，而是对系统的逻辑和物理视图不断细化的迭代开发过程。

Booch 方法的力量在于其丰富的符号体系，包括类图（类结构，静态视图）、对象图（对象结构，静态视图）、状态转移图（类结构，动态视图）、时态图（对象结构，动态视图）、模块图（模块体系结构）、进程图（进程体系结构）。

4.3.2　OMT 方法

Rumbaugh 的 OMT（Object Modeling Technology，对象建模技术）方法从三个视角描述系统，相应地提供了三种模型，分别是对象模型、动态模型和功能模型。

对象模型描述对象的静态结构和它们之间的关系，主要的概念包括类、属性、操作、继承、关联（即关系）、聚集；动态模型描述系统那些随时间变化的方面，主要概念有状态、子状态和超状态、事件、行为、活动；功能模型描述系统内部数据值的转换，主要概念有加工、数据存储、数据流、控制流、角色（源/潭）。

OMT 方法将开发过程分为 4 个阶段，分别是分析、系统设计、对象设计和实现。

（1）系统分析。基于问题和用户需求的描述，建立现实世界的模型。分析阶段的产物有问题描述、对象模型（对象图+数据词典）、动态模型（状态图+全局事件流图）、功能模型（数据流图+约束）。

（2）系统设计。结合问题域的知识和目标系统的体系结构（求解域），将目标系统分解为子系统。

（3）对象设计。基于分析模型和求解域中的体系结构等添加的实现细节，完成系统设计。主要产物包括细化的对象模型、动态模型和功能模型。

（4）系统实现。将系统设计转换为特定的编程语言或硬件，同时保持可追踪性、灵活性和可扩展性。

4.3.3 Coad/Yourdon 方法

Coad/Yourdon 方法严格区分了 OOA 和 OOD，利用 5 个层次的活动来定义和记录系统行为、输入/输出。这 5 个层次的活动包括：

（1）发现类及对象。描述如何发现类及对象。从应用领域开始识别类及对象，形成整个应用的基础，然后，据此分析系统的责任。

（2）识别结构。该阶段分为两个步骤。第一，识别一般-特殊结构，该结构获取了识别出的类的层次结构；第二，识别整体-部分结构，该结构用来表示一个对象如何成为另一个对象的一部分，以及多个对象如何组装成更大的对象。

（3）定义主题。主题由一组类及对象组成，用于将类及对象模型划分为更大的单位，便于理解。

（4）定义属性。其中包括定义类的实例（对象）之间的实例连接。

（5）定义服务。其中包括定义对象之间的消息连接。

在面向对象分析阶段，经过 5 个层次的活动后的结果是一个分成 5 个层次的问题域模型，包括主题、类及对象、结构、属性和服务 5 个层次，由类及对象图表示。5 个层次活动的顺序并不重要。

OOD 模型可进一步区分为 4 个部分，分别是问题域的设计、人机交互界面的设计、应用控制的设计、与问题域有关的设计。

1．问题域的设计

问题域部分的设计是任何 OOD 方法都必须完成的工作，它主要是对 OOA 结果进行改进和细化，并将其由问题域转化到解域，具体来说，有以下几个方面。

（1）属性：有些属性在分析阶段有助于问题的理解，而到了设计阶段则可以由其他属性导出或根本没必要保留。因此，应将去掉它们。相反地，为了实现服务算法还需要增加相应的一些属性。

（2）服务：OOA 只给出了服务的接口，其具体实现算法要在 OOD 阶段完成。

（3）类及对象：在 OOA 阶段有助于问题理解的一些类在 OOD 阶段成为冗余，需要删除；而为了优化、调整继承关系，又需要增加一些类。所有的类都确定以后，还要明确哪些类的对象会引发哪些类创建新对象。

（4）结构：对类之间的结构进行优化调整。

（5）对象行为：明确对象之间消息传递的实现算法，依据动态模型确定对象之间消息发送的先后顺序，并设计相应算法，协调对象的行为。

2. 人机交互界面的设计

有些设计方法并没有提到交互界面的设计，一方面是因为这些系统中交互界面不十分重要，另一方面是因为这部分的设计很有规律，设计方法也比较成熟。在 Coad/Yourdon 方法方法中，这部分的工作主要包括以下一些。

（1）交互界面子系统的设计：与界面有关的类及类之间结构的设计，以及有关算法的设计。

（2）交互界面子系统和应用之间接口的设计。

3. 应用控制的设计

这部分对象主要完成应用的驱动工作。这部分对象不同于从现实世界中抽象出来的对象，在现实世界和问题域中没有原型，它们与界面子系统中的对象发生作用，控制系统的运行。

4. 与问题域有关的设计

一些系统具有与应用领域有关的特点，例如，多任务、分布式计算等，该项工作主要是针对这些特点完成相应设计的。

4.3.4　Jacobson 方法

Jacobson 方法涉及到整个软件生命周期的多个阶段，包括需求分析、设计、实现和测试等。需求分析和设计密切相关。需求分析阶段的活动包括定义潜在的角色、识别问题域中的对象和关系、基于需求规范说明和角色的需要发现用例、详细描述用例；设计阶段包括从需求分析模型中发现设计对象和针对实现环境调整设计模型。

Jacobson 方法将对象区分为语义对象（领域对象）、界面对象（用户界面对象）和控制对象（处理界面对象和领域对象之间的控制）。该方法的一个关键概念就是用例。

4.3.5　设计基本原则

OOD 的一些基本准则是：模块化、抽象、信息隐藏、高内聚和低耦合，这些是与 SD 方法相似的。本节具体介绍一些 OOD（含设计模式）的原则，这些原则有助于软件设计师设计出具有弹性的系统，从而消除系统设计中存在的问题。

（1）单一职责原则：这是模块内聚性在类和类的职责中的体现，如果一个类承担的职责过多，意味着这些职责耦合在一起，形成的很有可能是一个“杂凑类”，任一个职责

的变化可能会削弱或者抑制该类完成其他职责的能力，并影响到构建、测试和部署等活动。通过业务分离可以对概念进行解耦，从而得到目的单一的类。

（2）开放-封闭原则。在模块本身不变动的情况下，通过改变模块周围的环境达到修改目的。遵循开放-封闭原则设计出的模块具有一个主要特征，即对于扩展是开放的，对于修改是封闭的。也就是说，模块的行为是可扩展的，当应用的需求改变时，在模块上进行扩展使其具有满足那些改变的新行为。当模块进行扩展时，不必改动模块的源代码或二进制代码。

（3）李氏（Liskov）替换原则。子类型必须能够替换掉它们的基类型；子类具有扩展父类的责任，而不是重写的责任。也就是说，基类的使用者不必为了使用子类而做任何其他的事情，他们可以在根本不了解子类的特殊性，甚至不必知道是否存在子类，存在哪些子类的情况下来调用基类的抽象方法。这样多态性才能顺利实现。事实上，正是Livkov 替换原则，才使开放-封闭原则得以实现。因为正是子类的可替换性才使得基类的模块在无需修改的情况下就可以扩展。

（4）依赖倒置原则。高层模块不应该依赖于低层模块，二者都应该依赖于抽象；抽象不应该依赖于细节，细节应该依赖于抽象。每个较高的层次都为它需要的服务声明一个抽象接口，较低的层次实现这个接口，每个高层类都通过该抽象接口使用下一层；要依赖于抽象，而不是具体实现。也可以这样说，要针对接口编程，不要针对实现编程。

（5）接口隔离原则。应当为客户端提供尽量小的单独的接口，而不是提供大的接口；使用多个专门的接口比使用单一的总接口要好。也就是说，一个类对另外一个类的依赖性应当是建立在最小的接口上的。这里的“接口”往往有两种不同的含义：一种是指一个类型所具有的方法特征的集合，仅仅是一种逻辑上的抽象；另外一种是指某种语言具体的“接口”定义，有严格的定义和结构。在进行 OOD 的时候，一个重要的工作就是恰当的划分角色和角色对应的接口。将没有关系的接口合并在一起，是对角色和接口的“污染”。如果将一些看上去差不多的接口合并，并认为这是一种代码优化，这也是错误的。不同的角色应该交给不同的接口，而不能都交给一个接口。

（6）组合重用原则：要尽量使用组合，而不是继承关系达到重用目的。组合相比于继承的好处：更大的灵活性而不会影响调用代码；更短的编译时间；适用性更广；较好的健壮性和安全性，更少的复杂性和脆弱性，更好的可维护性；能够在运行期创建新的对象。当然，耦合度的降低又会增加设计的难度、系统的复杂性以及实现的成本等，因此，在设计过程中，必须对这些因素综合考虑。

（7）迪米特（Demeter）原则：又叫最少知识法则（Least Knowledge Principle，LKP），就是说一个对象应当对其他对象有尽可能少的了解。遵循类之间的迪米特原则会使一个系统的局部设计简化，因为每一个局部都不会和远距离的对象有直接的关联。但是，这也会造成系统的不同模块之间的通信效率降低，也会使系统的不同模块之间不容易协调。迪米特原则在类的设计上主要体现在：优先考虑将类设置成不变类，尽量降低类的访问

权限，谨慎使用 Serializable，尽量降低成员的访问权限。但遵循该原则，会在系统中造出大量的小方法，这些方法仅仅是传递间接的调用，与系统的商务逻辑无关。

4.4　用户界面设计

美的界面能消除用户由感觉引起的乏味、紧张和疲劳（情绪低落），大大提高用户的工作效率，从而进一步为发挥用户技能和为用户完成任务做出贡献。从人机界面发展历史与趋势上可以看出，人们对界面美的需求，以及美在界面设计中的导向作用。

4.4.1　用户界面的特点

界面设计已经经历了两个界限分明的时代。第一代是以文本为基础的简单交互，如常见的命令行、字符菜单等。由于第一代界面考虑人的因素太少，用户兴趣不高。随着技术的发展，出现了第二代直接操纵的界面。它大量使用图形、语音和其他交互媒介，充分地考虑了人对美的需求。直接操纵的界面使用视听、触摸等技术，让人可以凭借生活常识、经历和推理来操纵软件，愉快地完成任务。更高层次的界面甚至模拟了人的生活空间，例如虚拟现实环境。

界面的美充分体现了人机交互作用中人的特性与意图，越来越多的用户将通过具有吸引力而令人愉快的人机界面与计算机打交道。

就一般的系统而言，一个好的用户界面应具有以下特点。

（1）可使用性：包括使用的简单性、用户界面中所用术语的标准化和一致性、拥有 HELP 帮助功能、快速的系统响应和低的系统成本、用户界面应具有容错能力。

（2）灵活性：考虑用户的特点、能力、知识水平，应当使用户界面能够满足不同用户的要求；用户可以根据需要制定和修改界面方式；系统能够按照用户的希望和需要，与其他软件系统有标准的界面。

（3）复杂性和可靠性：用户界面的规模和组织的复杂程度就是界面的复杂性，用户界面的可靠性是指无故障使用的间隔时间。

4.4.2　设计原则

用户界面设计是一项复杂的任务，它必须遵循一些“良好设计”的指导原则，以下介绍一些关键的用户界面设计原则。

（1）用户控制。人机界面设计首先要确立用户类型。划分类型可以从不同的角度，视实际情况而定。确定类型后要针对其特点预测他们对不同界面的反应。这就要从多方面设计分析。用户应当感觉系统的运行在自己的控制之下。在图形界面或基于 Web 的界面中，用户指导程序的每一步执行；即使在程序进行某些处理或用户等待输出结果时，

用户同样保持对控制的敏感度。

（2）信息最小量。人机界面设计要尽量减少用户记忆负担，采用有助于记忆的设计方案。

（3）帮助和提示。要对用户的操作命令做出反应，帮助用户处理问题。系统要设计有恢复出错现场的能力，在系统内部处理工作要有提示，尽量把主动权让给用户。

（4）媒体最佳组合。多媒体界面的成功并不在于仅向用户提供丰富的媒体，而应在相关理论指导下，注意处理好各种媒体间的关系，恰当选用。

（5）界面一致性。一致性要求用户界面遵循标准和常规的方式，让用户处在一个熟悉的和可预见的环境之中，这主要体现在命名、编码、缩写、布局以及菜单、按钮和键盘功能在内的控制使用等。

（6）界面容错性。一个好的界面应该以一种宽容的态度允许用户进行实验和出错，使用户在出现错误时能够方便地从错误中恢复。

（7）界面美观性。界面美观性是视觉上的吸引力，主要体现在具有平衡和对称性、合适的色彩、各元素具有合理的对齐方式和间隔、相关元素适当分组、使用户可以方便地找到要操作的元素等。

（8）界面可适应性。界面可适应性是指用户界面应该根据用户的个性要求及其对界面的熟知程度而改变，即满足定制化和个性化的要求。所谓定制化是在程序中声明用户的熟知程度，用户界面可以根据熟知程度改变外观和行为；所谓个性化是使用户按照自己的习惯和爱好设置用户界面元素。

4.5　设计评审

在设计阶段结束时要进行严格的技术评审，尽量不让错误传播到下一个阶段。设计评审一般采用评审会议的形式来进行。软件设计评审流程如图 4-13 所示。

（1）设计负责人：一般工程设计均由软件公司选派设计负责人，设计负责人承担该项工程的全部设计管理任务。对设计质量、进度及各单项设计间的组织协调等全面负责，对各单项工程之间的衔接、协调和总体方案质量负主要责任，并负责编写总说明，汇编总概算。

（2）高级管理人员：确定主审员、审批评审记录。

（3）主审员：在评审会前提出项目的书面评审意见，确定评审组，确定评审结果并填写评审记录。

（4）评审组：专业评审组评委表决通过项目初评结论，并报综合评审会议；通过设计报告。

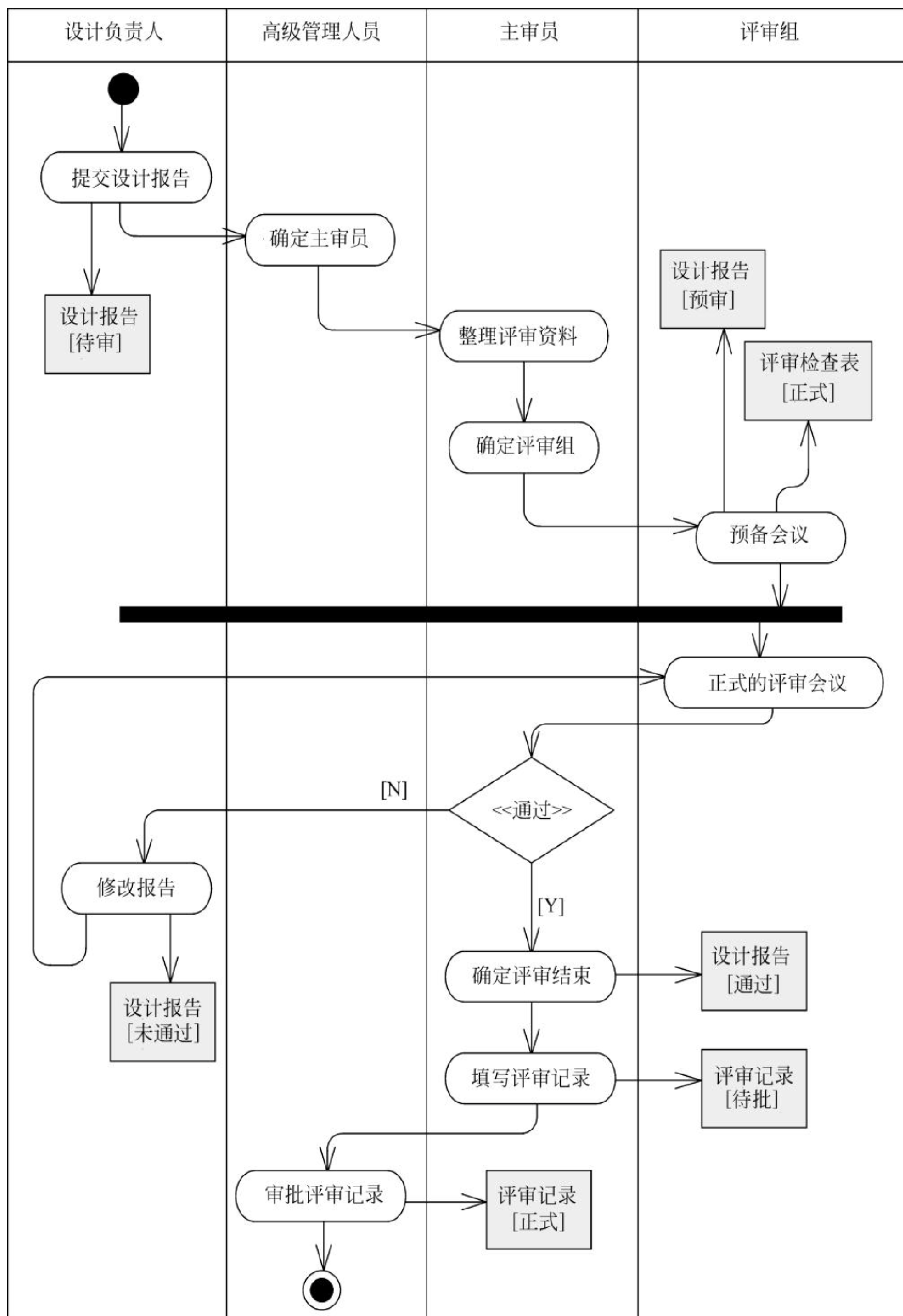


图 4-13　软件设计评审流程图

本章参考文献

[1] 张家重，李刚，郑向伟. 从 OOA 到 OOD. 小型微型计算机系统，1997

[2] 孙光明，徐宝祥，刘凤勤. OOA 在信息系统概念建模中的概念模型方法研究. 现代情报，1999

[3] 白庆华. 传统的结构化分析方法之弊端及 OOA 的出现. 计算机系统应用，1994

[4] 郑人杰，殷人昆，陶永雷. 实用软件工程. 北京：清华大学出版社，2001

[5] 梅宏. 软件工程——实践者的研究方法. 第 5 版. 北京：机械工业出版社，2002

[6] 陈小群. 面向对象的系统分析设计. http://www.csai.cn

第5章　软件测试

软件测试是为了找出程序中的错误而执行程序的过程，根据程序开发阶段的规格说明及程序内部结构，精心设计一批测试用例（输入数据及其预期结果的集合），并利用这些测试用例去运行程序，以发现程序错误的过程。

Myers 认为，一个好的测试用例在于能发现至今未发现的错误，一个成功的测试是发现了至今未发现的错误的测试。换言之，测试的目的是：

（1）想以最少的时间和人力，系统地找出软件中潜在的各种错误和缺陷。

（2）测试的附带收获是，它能够证明软件的功能和性能与需求说明相符合。

（3）测试过程中收集到的测试结果数据为可靠性分析提供了依据。

（4）测试不能表明软件中不存在错误，它只能说明软件中存在错误。

应当把“尽早地和不断地进行测试”作为软件开发人员的座右铭，软件测试的一些基本原则如下：

（1）程序员应避免检查自己的程序。

（2）充分注意测试中的群集现象。经验表明，测试后程序中残存的错误数目与该程序中已发现的错误数目成正比。

（3）严格执行测试计划，排除测试的随意性。

（4）应当对每一个测试结果做全面检查。

（5）妥善保存测试计划、测试用例、出错统计和最终分析报告，为软件维护提供方便。

希赛教育专家提示：软件测试并不等于程序测试。软件测试应贯穿于软件定义与开发的整个期间。需求分析、概要设计、详细设计，以及程序编码等各阶段所得到的文档，包括需求规格说明、概要设计说明、详细设计说明以及源程序，都应成为软件测试的对象。

5.1　测试用例设计

测试用例是为特定目标开发的测试输入、执行条件和预期结果的集合。测试用例应当由测试输入数据和对应的预期输出结果这两部分组成，在设计测试用例时，应包括合理的输入条件和不合理的输入条件。

5.1.1　黑盒测试

黑盒测试又称为功能测试或数据驱动测试，它把测试对象看作为一个黑盒子，不考

虑程序的内部逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符合它的功能说明。

黑盒测试方法主要是在程序的接口上进行测试，主要是为了发现以下错误：

（1）是否有不正确或遗漏了的功能。

（2）在接口上，能否正确的接收输入，能否输出正确的结果。

（3）是否有数据结构错误或外部信息访问错误。

（4）性能上是否能够满足要求。

（5）是否有初始化或终止性错误。

黑盒测试需要在所有可能的输入条件和输出条件中确定测试数据，来检查程序是否都能产生正确的输出。有时测试数据量太大，设计全部路径覆盖的测试数据是不现实的。黑盒测试的测试用例设计方法主要有等价类划分、边界值分析、错误推测、因果图等。

1．等价类划分

等价类划分是一种典型的黑盒测试方法，使用这一方法时，完全不考虑程序的内部结构，只依据程序的规格说明来设计测试用例。该方法首先把所有可能的输入数据（即程序的输入域）划分为若干个部分，然后，从每一部分中选取少数有代表性的数据作为测试用例。使用这一方法设计测试用例要经历划分等价类（列出等价类表）和选取测试用例两个步骤。

第一步，划分等价类。等价类是指某个输入域的子集合。在该子集合中，各个输入数据对于揭露程序中的错误都是等效的。也就是说，测试某等价类的代表值就等价于对这一类其他值的测试。等价类的划分有两种不同的情况，分别是有效等价类和无效等价类。有效等价类是指对于程序的规格说明来说，是合理的、有意义的输入数据构成的集合；无效等价类是指对于程序的规格说明来说，是不合理的、无意义的输入数据构成的集合。

第二步，从划分出的等价类中按以下原则选择测试用例：

（1）为每一个等价类规定一个唯一编号。

（2）设计一个新的测试用例，使其尽可能多地覆盖尚未被覆盖的有效等价类，重复这一步，直到所有的有效等价类被覆盖为止。

（3）设计一个新的测试用例，使其只覆盖一个尚未被覆盖的无效等价类，重复此步骤，直到所有的无效等价类都覆盖为止。

2．边界值分析

边界值分析也是一种黑盒测试方法，是对等价类划分方法的补充。人们从长期的测试工作经验得知，大量的错误是发生在输入或输出范围的边界上，而不是在输入范围的内部。因此，针对各种边界情况设计测试用例，可以查出更多的错误。使用边界值分析设计测试用例，应当选取正好等于、刚刚大于、刚刚小于边界的值作为测试数据。例如，对一个取值在 1～8 范围内的变量设计测试用例时，则可以取 0，1，8，9。

3. 错误推测法

人们也可以靠经验和直觉推测程序中可能存在的各种错误，从而有针对性地编写检查这些错误的例子。其基本思想是：列举出程序中所有可能的错误和容易发生错误的特殊情况，根据它们选择测试用例。

4. 因果图

如果在测试时必须考虑输入条件的各种组合，可使用一种适合于描述对于多种条件的组合，相应产生多个动作的形式来设计测试用例，这就需要利用因果图。这种方法最终生成的就是判定表，它适合于检查程序输入条件的各种组合情况。用因果图生成测试用例的基本步骤是：

（1）分析软件需求规格说明描述中，哪些是原因（即输入条件或输入条件的等价类），哪些是结果（即输出条件），并给每个原因和结果赋予一个标识符。

（2）分析软件规格说明描述中的语义，找出原因与结果之间、原因与原因之间对应的是什么关系，根据这些关系，画出因果图。

（3）由于语法或环境限制，有些原因与原因之间、原因与结果之间的组合情况不可能出现。为表明这些特殊情况，在因果图上用一些记号标明约束或限制条件。

（4）把因果图转换成判定表。

（5）把判定表的每一列拿出来作为依据，设计测试用例。

5.1.2　白盒测试

白盒测试又称为结构测试或逻辑驱动测试，它把测试对象看作一个透明的盒子，允许测试人员利用程序内部的逻辑结构和有关信息，设计测试用例，对程序中所有逻辑路径进行测试。通过在不同点检查程序的状态，确定实际的状态是否与预期的状态一致。

白盒测试主要对程序模块进行如下检查：

（1）对程序模块的所有独立的执行路径至少测试一次。

（2）对所有的逻辑判定，取“真”与取“假”的两种情况都至少测试一次。

（3）在循环的边界和运行界限内执行循环体。

（4）测试内部数据结构的有效性。

白盒测试用例的一种典型设计技术是逻辑覆盖，包括语句覆盖、判定覆盖、条件覆盖、判定-条件覆盖、条件组合覆盖、路径覆盖等。

（1）语句覆盖：设计若干个测试用例，运行被测程序，使得每一条可执行语句至少执行一次。

（2）判定覆盖（分支覆盖）：设计若干个测试用例，运行被测程序，使程序中每个判断的取真分支和取假分支至少经历一次。

（3）条件覆盖：设计若干个测试用例，运行被测程序，使得程序中每个判断的每个

条件的可能取值至少执行一次。

（4）判定-条件覆盖：设计足够的测试用例，使得判断中每个条件的所有可能取值至少执行一次，同时每个判断中的所有可能判断结果至少执行一次。

（5）条件组合覆盖：设计足够的测试用例，运行被测程序，使得每个判断的所有可能的条件取值组合至少执行一次。

（6）路径覆盖：设计足够的测试用例，覆盖程序中所有可能的路径。

5.2 软件测试的步骤

从测试实际的前后过程来看，软件测试是由一系列不同的测试所组成的，这些测试的步骤可分为单元测试（模块测试）、集成测试（组装测试）、确认测试和系统测试。软件开发的过程是自顶向下的，测试则正好相反，以上这些过程是自底向上，逐步集成的。

1．单元测试

单元测试是针对每个模块进行的测试，可从程序的内部结构出发设计测试用例，多个模块可以平行测试。单元测试通常在编码阶段进行，必要的时候要编写驱动模块和桩模块。驱动模块是指在单元测试和集成测试中，协调输入和输出的测试程序；桩模块指模拟被调用单元的程序。

单元测试可以测试：模块接口、局部数据结构、边界条件、独立路径和错误处理路径 5 个方面的内容。

（1）模块接口测试：调用本模块的输入参数是否正确；本模块调用子模块时输入给子模块的参数是否正确；全局变量的定义在各模块中是否一致；在做内外存交换时要考虑文件属性是否正确，Open 与 Close 语句是否正确；缓冲区容量与记录长度是否匹配，在进行读写操作之前是否打开了文件，在结束文件处理时是否关闭了文件；I/O 错误是否作了检查并做了处理，动态申请的内存是否释放等。

（2）局域数据结构测试：不正确或不一致的数据类型说明，使用尚未赋值或尚未初始化的变量，错误的初始值或错误的默认值；变量名拼写错或书写错，不一致的数据类型；全局数据对模块的影响，是否非法对空指针操作，数组下标和内存操作是否越界等。

（3）路径测试：选择适当的测试用例，对模块中重要的执行路径进行测试、应当设计测试用例查找由于错误的计算、不正确的比较或不正常的控制流而导致的错误、对基本执行路径和循环进行测试可以发现大量的路径错误。

（4）错误处理测试：出错的描述是否难以理解，是否能够对错误定位，显示的错误与实际的错误是否相符；对错误条件的处理正确与否，在对错误进行处理之前错误条件是否已经引起系统的干预；函数入口是否作了指针参数、是否为空，以及其他参数的正确性检查等。

（5）边界测试：注意数据流、控制流中刚好等于、大于或小于确定的比较值时出错

的可能性。对这些地方要仔细地选择测试用例，认真加以测试。如果对模块运行时间有要求的话，还要专门进行关键路径测试，以确定最坏情况下和平均意义下影响模块运行时间的因素。

2．集成测试

在单元测试的基础上，将所有模块按照概要设计要求组装成为系统，必须精心计划，应提交集成测试计划、集成测试规格说明和集成测试分析报告。

这时需要考虑的问题是：

（1）在把各个模块连接起来的时候，穿越模块接口的数据是否会丢失。

（2）一个模块的功能是否会对另一个模块的功能产生不利的影响。

（3）各个子功能组合起来，能否达到预期要求的父功能。

（4）全局数据结构是否有问题。

（5）单个模块的误差累积起来，是否会放大，从而达到不能接受的程度。

把模块组装成为系统的方式有两种，分别是一次性组装方式和增殖式组装方式。在组装测试时，应当确定关键模块，对这些关键模块及早进行测试，这些模块有如下特征：满足某些软件需求，在程序的模块结构中位于较高的层次，较复杂易发生错误，有明确定义的性能要求。

希赛教育专家提示：软件集成的过程是一个持续的过程，会形成多个临时版本。在不断地集成过程中，功能集成的稳定性是真正的挑战。在每个版本提交时，都需要进行“冒烟”测试，即对程序主要功能进行验证。冒烟测试也称为版本验证测试或提交测试。

3．确认测试

确认测试验证软件的功能、性能，以及其他特性是否与用户的需求规格说明一致。

对软件的功能和性能要求在软件需求规格说明书中已经明确规定，它包含的信息就是软件确认测试的基础。确认测试的主要步骤为：

首先，进行有效性测试。在开发环境或实际运行环境下，运用黑盒测试的方法，验证被测软件是否满足需求规格说明书列出的功能和性能需求。通过制定确认测试计划和执行测试计划进行，验证软件的功能和性能等特性是否与需求相符，所有的文档是否正确且便于使用。同时，对于其他软件需求（可移植性、兼容性、出错自动恢复、可维护性等）都要进行测试。

其次，进行软件配置复查。保证做到软件配置的所有成分都齐全，各方面的质量都符合要求，具有维护阶段所必需的细节，而且已经编排好分类的目录。

最后，进行验收测试。验收测试是以用户为主的测试，软件开发人员和质量保证人员也应参加，由用户参加设计测试用例，使用生产中的实际数据进行测试；在测试过程中，除了考虑软件的功能和性能外，还应对软件的可移植性、兼容性、可维护性、错误的恢复功能等进行确认。

4．系统测试

如果项目不只包含软件，还有硬件和网络等，则要将软件与外部支持的硬件、外设、支持软件、数据等其他系统元素结合在一起，在实际运行环境下，对计算机系统进行一系列集成与确认测试。一般来说，系统测试的主要内容包括功能测试、健壮性测试、性能测试、用户界面测试、安全性测试、安装与反安装测试等。系统测试计划通常在系统分析阶段（需求分析阶段）完成。

5．α 测试和 β 测试

在软件交付使用后，用户将如何实际使用程序，对于开发者来说是不知道的。因此，通常在软件发布上市之前进行 α 测试和 β 测试。

α 测试是由用户在开发环境下进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的测试。α 测试的目的是评价软件产品的功能、局域化、可使用性、可靠性、性能和支持，尤其注重产品的用户界面和特色。α 测试可以从软件产品编码结束之时开始，或在模块（子系统）测试完成之后开始，也可以在确认测试过程中产品达到一定的稳定和可靠程度之后再开始。

β 测试是由软件的多个用户在实际使用环境下进行的测试。这些用户返回有关错误信息给开发者。β 测试时，开发者通常并不在测试现场。因此，β 测试是在开发者无法控制的环境下进行的软件现场应用。在 β 测试中，由用户记下遇到的所有问题，包括真实的以及主观认定的，定期向开发者报告。β 测试着重于产品的支持性，包括文档、客户培训和支持产品生产能力。只有当 α 测试达到一定的可靠程度时，才能开始 β 测试。它处在整个测试的最后阶段。同时，产品的所有手册文本也应该在此阶段完全定稿。

5.3 软件测试种类

软件测试是由一系列不同的测试组成，主要目的是对以计算机为基础的系统进行充分的测试。

（1）功能测试。功能测试是在规定的一段时间内运行软件系统的所有功能，以验证这个软件系统有无严重错误。

（2）可靠性测试。如果系统需求说明书中有对可靠性的要求，则需进行可靠性测试。有关可靠性的评价指标和方法，请阅读第 7 章。

（3）强度测试。强度测试是要检查在系统运行环境不正常乃至发生故障的情况下，系统可以运行到何种程度的测试。例如，把输入数据速率提高一个数量级，确定输入功能将如何响应；设计需要占用最大存储量或其他资源的测试用例进行测试；设计在虚拟存储管理机制中引起“颠簸”的测试用例进行测试；设计会对磁盘常驻内存的数据过度访问的测试用例进行测试。强度测试的一个变种就是敏感性测试。在程序有效数据界限内一个小范围内的一组数据可能引起极端的或不平稳的错误处理出现，或者导致极度的

性能下降的情况发生。强度测试是用以发现可能引起这种不稳定性或不正常处理的某些数据组合。

（4）性能测试。性能测试是要检查系统是否满足在需求说明书中规定的性能。特别是对于实时系统或嵌入式系统；性能测试常常需要与强度测试结合起来进行，并常常要求同时进行硬件和软件检测；通常，对软件性能的检测表现在以下几个方面：响应时间、吞吐量、辅助存储区，例如缓冲区，工作区的大小等、数据处理精度等。

（5）恢复测试。恢复测试是要证实在克服硬件故障（包括掉电、硬件或网络出错等）后，系统能否正常地继续进行工作，并不对系统造成任何损害；为此，可采用各种人工干预的手段，模拟硬件故障，故意造成软件出错。并由此检查错误探测功能：系统能否发现硬件失效与故障；能否切换或启动备用的硬件；在故障发生时能否保护正在运行的作业和系统状态；在系统恢复后能否从最后记录下来的无错误状态开始继续执行作业，等等；掉电测试：其目的是测试软件系统在发生电源中断时能否保护当时的状态且不毁坏数据，然后在电源恢复时从保留的断点处重新进行操作。

（6）启动/停止测试。这类测试的目的是验证在机器启动及关机阶段，软件系统正确处理的能力。这类测试包括：反复启动软件系统（例如，操作系统自举、网络的启动、应用程序的调用等）；在尽可能多的情况下关机。

（7）配置测试。这类测试是要检查计算机系统内各个设备或各种资源之间的相互联结和功能分配中的错误。它主要包括以下几种：配置命令测试即验证全部配置命令的可操作性（有效性），特别是对最大配置和最小配置要进行测试，软件配置和硬件配置都要测试；循环配置测试即证明对每个设备物理与逻辑的，逻辑与功能的每次循环置换都能正常工作；修复测试即检查每种配置状态及哪个设备是坏的，并用自动的或手工的方式进行配置状态间的转换。

（8）安全性测试。安全性测试是要检验在系统中已经存在的系统安全性、保密性措施是否发挥作用，有无漏洞。力图破坏系统的保护机构以进入系统的主要方法有以下几种：正面攻击或从侧面、背面攻击系统中易受损坏的那些部分；以系统输入为突破口，利用输入的容错性进行正面攻击；申请和占用过多的资源压跨系统，以破坏安全措施，从而进入系统；故意使系统出错，利用系统恢复的过程，窃取用户口令及其他有用的信息；通过浏览残留在计算机各种资源中的垃圾（无用信息），以获取如口令、安全码、译码关键字等信息；浏览全局数据，期望从中找到进入系统的关键字；浏览那些逻辑上不存在，但物理上还存在的各种记录和资料等。

（9）可使用性测试。可使用性测试主要从使用的合理性和方便性等角度对软件系统进行检查，发现人为因素或使用上的问题。要保证在足够详细的程度下，用户界面便于使用；对输入量可容错、响应时间和响应方式合理可行、输出信息有意义、正确并前后一致；出错信息能够引导用户去解决问题；软件文档全面、正规、确切。

（10）安装测试。安装测试的目的不是查找软件错误，而是查找安装错误。在安装

软件系统时，会有多种选择，例如要分配和装入文件与程序库、布置适用的硬件配置以及进行程序的联结；而安装测试就是要找出在这些安装过程中出现的错误。安装测试是在系统安装之后进行测试。它要检验：用户选择的一套任选方案是否相容，系统的每一部分是否都齐全，所有文件是否都已产生并确有所需要的内容，硬件的配置是否合理，等等。

（11）过程测试。在一些大型的系统中，部分工作由软件自动完成，其他工作则需由各种人员，包括操作员、数据库管理员、终端用户等，按一定规程同计算机配合，靠人工来完成。指定由人工完成的过程也需要经过仔细的检查，这就是所谓的过程测试。

（12）容量测试。容量测试是要检验系统的能力最高能达到什么程度，例如：对于编译程序，让它处理特别长的源程序，对于操作系统，让它的作业队列“满员”，对于信息检索系统，让它使用频率达到最大。在使系统的全部资源达到“满负荷”的情形下，测试系统的承受能力。

（13）文档测试。这种文档测试是检查用户文档（如用户手册）的清晰性和精确性。

（14）兼容性测试。这类测试主要想验证软件产品在不同版本之间的兼容性。有两类基本的兼容性测试：向下兼容和交错兼容。

希赛教育专家提示：以上介绍的测试种类是从不同的侧面来描述测试工作的，其中含有重叠的部分。例如，可以把可靠性测试看作是性能测试的一部分。

5.4 软件测试自动化工具

软件测试自动化工具包括测试数据自动生成程序、静态分析程序、动态分析程序、测试结果分析程序，以及驱动测试的测试数据库等。

软件测试的工作量很大，据统计，会占到40%左右的开发时间。一些可靠性要求非常高的软件，测试时间甚至占到总开发时间的60%。但测试却是在整个软件开发过程中极有可能应用计算机进行自动化工作的阶段，原因是测试的许多操作是重复性的、非智力创造性的工作，计算机就最适合于代替人类去完成这些任务。企业在软件测试自动化工具的投资，会对整个开发工作的质量、成本和周期带来非常明显的效果。

一些适于进行自动化的测试操作为：

（1）测试用例的生成，包括测试输入、标准输出、测试操作指令等。

（2）测试的执行控制，包括单机与网络多机分布运行，夜间及假日运行，测试用例调用控制；测试对象、范围、版本控制等。

（3）测试结果与标准输出的对比。

（4）不吻合的测试结果的分析、记录、分类和通报。

（5）测试结果的汇总统计和报表的产生。

测试自动化与软件配置管理是密不可分的。与测试有关的资源都应在配置管理中进

行统一考虑。另外，测试工具的采用也是一个提高质量的关键，有些专用的测试工具能帮助用户发现一些用任何测试用例都难以触及的错误。

5.4.1　白盒测试工具

白盒测试工具主要是用于代码开发阶段，检查应用的可靠性和稳定性。本节以 Windows 平台中的 NuMega DecPartner Studio 工具为例，介绍白盒测试工具的功能。

NuMega DecPartner Studio 满足在软件开发过程中每一个开发人员的需求，无论是使用一种或多种语言，该工具都能够帮助开发团队提高生产力，主要有自动的错误检测、性能分析、代码覆盖分析等功能，分别用于获取、定位错误，抽取代码执行频度，以及抽取代码覆盖率等数据。其产品包括 BoundsChecker、TrueCoverage、TrueTime、SmartCheck、CodeReview。

1．BoundsChecker

程序员在开发过程中可能会经常遇到这样的问题：调试时语法没有问题，代码也没有错误，但应用程序运行就是不正常甚至死机。其实，这有可能是由于逻辑错误引起的内存溢出或资源泄露等问题，这些错误一般是不容易被检测出来的。而这类错误就是 BoundsChecker 错误检测范围之一。

通过对被测应用程序的操作，BoundsChecker 提供清晰的、详细的程序错误分析，自动查明静态的堆栈错误及内存/资源泄露，并能够迅速定位出错的源代码，即使在没有源代码的情况下也可检查第三方构件的错误。

BoundsChecker 错误检测范围主要包括以下一些错误。

（1）指针和泄露错误：接口泄露、内存泄露、资源泄露、未分配的指针错误。

（2）内存错误：动态存储溢出、无效的句柄被锁定、句柄没有被锁定、内存分配冲突、栈空间溢出、静态存储溢出。

（3）API（Application Programming Interface，应用程序接口）和 OLE 错误：API 函数返回失败、API 函数未执行、无效的变量（包括指针变量、字符串变量等）、OLE 接口方法的变量无、OLE 接口方法失败、线程调用库函数错误。

BoundsChecker 支持的语言有 C++和 Delphi，支持的主机平台有 Windows 系统。

2．TrueCoverage

在开发过程中，对一个应用程序通过手工测试，总会有一部分代码功能没有被检测到，或者说逐个检测每一个函数的调用是相当费时间的。不能保证未被检测的代码的可靠性，以后程序的失败可能往往就是由这部分未检测的代码造成的。可以用 TrueCoverage 来帮助解决这些问题，在测试程序时，每完成一次应用对话，TrueCoverage 就能够列出在这次对话中所有函数被调用次数、所占比率等，并可以直接定位到源代码。当然，也可以合并多个应用对话来进行检测。TrueCoverage 能通过衡量和跟踪代码执行及代码稳定性，帮助开发团队节省时间和改善代码可靠性。

TrueCoverage 支持的语言有 C++、Java 和 Visual Basic，支持的主机平台有 Windows 系统。

3．TrueTime

代码运行缓慢是开发过程中的一个重要问题。应用程序运行速度较慢，程序员不容易找到到底是在哪里出现了问题。如果不能解决应用程序的性能，将降低并极大地影响应用程序的质量，因此，查找和修改性能瓶颈是调整整个代码性能的关键。当在测试程序时，每完成一次应用对话，TrueTime 都能提供这次对话中函数的调用时间，提供详细的应用程序和构件性能的分析，并自动定位到运行缓慢的代码。这样，就能帮助程序员尽快地调整应用程序的性能。

TrueTime 支持的语言有 C++、Java 和 Visual Basic，支持的主机平台有 Windows 系统。

4．SmartCheck

作为一名 Visual Basic 的开发人员，在开发的过程中经常会遇到许多问题难以解决，包括隐藏的运行时错误、Windows API 函数在 Visual Basic 中正确使用的问题、一些构件的错误等，它们很难被定位到具体的代码中，令开发人员花费大量时间去寻找并解决。SmartCheck 就是能很快地查找到这些问题的一个自动化工具，它检测所有的 Windows API 函数调用、内存分配，以及其他一些重要的程序错误。SmartCheck 检错的种类包括泄露、接口方法失败、存储错误、程序和函数失败和 Visual Basic 的 Runtime 错误等，它能够将检测到的错误快速地定位到源代码。

5．CodeReview

对于 Visual Basic 开发人员来说，CodeReview 是较好的自动源代码分析工具，它对应用程序的构件、逻辑、Windows 和 Visual Basic 自身潜在的数百个问题进行严格的源代码检查。CodeReview 分析的类型包括逻辑错误、应用程序性能和可用性问题、Windows API 调用和标准一致性问题等。CodeReview 可以检测整个的 Visual Basic 工程或指定的模块，并能定制检错的种类。对检测的结果有详细的说明，提供帮助和推荐解决方案，而且能够直接地链接到源代码。CodeReview 系统还提供了两个子模块。一个是 Metrics，通过对 Visual Basic 工程的执行，计算出代码的长度、复杂度、理解度、语言的使用等级、出错的可能性等数据；另一个是 Namer，它调用一个 Visual Basic 工程，自动并规则地对其中的对象重新命名，并备份原来没有规则命名的工程文件，使开发人员对程序能够有条理地管理。

5.4.2 静态代码检查工具

C/C++语言的语法拥有其他语言所没有的灵活性，这种灵活性带来了代码效率的提升，但相应也使得代码编写具有很大的随意性。另外，C/C++编译器不进行强制类型检查，也不做任何边界检查，这就增加了代码中存在隐患的可能性。如果能够在代码提交

测试之前发现这些潜在的错误，就能够极大地减轻测试人员的压力，减少软件项目的除错成本。但是，传统的 C/C++编译器对此已经无能为力，这个任务只能由专用的代码检查工具完成。

目前，有很多 C/C++静态代码检查工具，其中 Logiscope RuleChecker 和 PC-Lint 是应用比较广泛的两个工具。这两个检查工具各有特色，Logiscope RuleChecker 倾向于代码编码规范的检查，例如，代码缩进格式、case 语句书写规范、函数声明和布尔表达式的编写规则等；而 PC-Lint 则偏重于代码的逻辑分析，它能够发现代码中潜在的错误，例如，数组和内存访问越界、内存泄漏、使用未初始化变量等。PC-Lint 代码检查工具可以很方便地与常见的代码编辑软件（如 Visual C++、Source Insight 等）集成。

PC-Lint 的全称是 PC-Lint/FlexeLint for C/C++，PC-Lint 能够在 Windows、MS-DOS 和 OS/2 平台上使用，以二进制可执行文件的形式发布；而 FlexeLint 运行于其他平台，以源代码的形式发布。PC-Lint 不仅能够对程序进行全局分析，识别没有被适当检验的数组下标，报告未被初始化的变量，警告使用空指针以及冗余的代码，还能够有效地帮程序员提出许多程序在空间利用、运行效率上的改进点。

通过下面的例子就可以看出 PC-Lint 工具的强大功能。

```
1:
2: char *report( int m, int n, char *p )
3: {
4:  int result;
5:  char *temp;
6:  long nm;
7:  int i, k, kk;
8:  char name[11] = "Joe Jakeson";
9:
10:  nm = n * m;
11:  temp = p == "" ? "null" : p;
12:  for( i = 0; i<m; i++ ) {
14:  k++;
15:  kk = i;
16:  }
17:
18:  if( k== 1 ) result = nm;
19:  else if( kk > 0 ) result = 1;
20:  else if( kk < 0 ) result = -1;
21:
22:  if( m == result ) return( temp );
23:  else return( name );
24: }
```

这是一段 C 代码，可以通过大多数常见的 C 语言编译器的检查，但是 PC-Lint 能够发现其中的错误和潜在的问题。第 8 行向 name 数组赋值时丢掉了结尾的“n”字符，第 10 行的乘法精度会失准，即使考虑到 long 比 int 的字长更长，由于符号位的原因仍然会造成精度失准，第 11 行的比较有问题，第 14 行的变量 k 没有初始化，第 15 行的 kk 可能没有被初始化，第 22 行的 result 也有可能没有被初始化，第 23 行返回的是一个局部对象的地址。

随着 C++语言的出现，C/C++编译器有了更严格的语法检查。但是，仍然不能避免出现有错误的程序。C++的类型检查依然不如 Pascal 那么严格。对于一个小程序，多数程序员都能够及时发现变量没有初始化、数组下标越界等错误，但是从一个拥有成千上万行代码的大型软件中找出这些瑕疵将是一项繁琐的工作，而且没有人可以保证能找出所有的这类问题。如果使用 PC-Lint，只需通过一次简单的编译就可以检查出这些错误，这将节省了大量的开发时间。从某种意义上说。PC-Lint 是一种更加严格的编译器，它除了可以检查出一般的语法错误外，还可以检查出那些虽然符合语法要求，但很可能是潜在的、不易发现的错误。

PC-Lint 能够检查出很多语法错误和语法上正确的逻辑错误，PC-Lint 为大部分错误消息都分配了一个错误号，编号小于 1000 的错误号是分配给 C 语言的，编号大于 1000 的错误号则用来说明 C++的错误消息。

PC-Lint 的检查分很多种类，有强类型检查、变量值跟踪、语义信息、赋值顺序检查、弱定义检查、格式检查、缩进检查、const 变量检查和 volatile 变量检查等。对每一种检查类型，PC-Lint 都有很多详细的选项，用以控制 PC-Lint 的检查效果。

5.4.3 黑盒测试工具

本节以 QACenter 为例，介绍黑盒测试工具的功能。

QACenter 帮助所有的测试人员创建一个快速、可重用的测试过程。这些测试工具自动帮助管理测试过程，快速分析和调试程序，包括针对回归、强度、单元、并发、集成、移植、容量和负载建立测试用例，自动执行测试和产生文档结果。QACenter 主要包括 QARun、QALoad、QADirector、TrackRecord、EcoTools。下面主要介绍 QARun 和 QALoad。

1．功能测试工具

在 QACenter 测试产品套件中，QARun 主要用于 C/S 应用客户端的功能测试，包括对图形用户界面的测试和客户端事物逻辑的测试。QARun 测试的实现方式是通过鼠标移动、键盘单击得到相应的测试脚本，对该脚本可以进行编辑和调试。在记录的过程中可针对被测试程序中所包含的功能点进行基线值的建立，换句话说，就是在插入检查点的同时建立期望值。通常，检查点在 QARun 提示目标系统执行一系列事件之后被执行。检查点用于确定实际结果与期望结果是否相同。

2．性能测试工具

QALoad 是企业范围的负载测试工具，该工具支持范围广，测试内容多，可以帮助软件测试人员、开发人员和系统管理人员对于分布式的应用执行有效的负载测试。负载测试能够模拟大批量用户的活动，从而发现大量用户负载下对 C/S 系统的影响。

5.4.4　内存问题动态检查工具

程序执行时内存错误和泄漏是最重要，也是最难寻找和修正的代码错误，其症状往往和导致内存错误的本质相去甚远，错误往往是随机事件的出现被触发。这样，程序平时看起来执行情况良好，却无法避免意外的发生。

Rational Purify 是一个内存问题动态检查的工具，主要用于开发阶段的白盒测试，是综合性检测运行时错误的工具，并可以和其他复合应用程序（包括多线程和多进程程序）一起工作。Purify 可以和 Microsoft Visual Studio 自动整合并且无需特殊的编译，用户可以在不改变工作方式的前提下使用 Purify。

Purify 可以帮助用户完成的任务有以下一些：

（1）对于 VC 程序的快捷的，易于理解的执行错误检测。

（2）即使没有代码也可以实现错误检查。

（3）代码覆盖寻找未测试代码部分。

Purify 检查每一个内存操作，定位错误发生的地点并提供尽可能详细的信息帮助程序员分析错误发生的原因。

它可以发现的主要错误有数组读写越界、使用未初始化的内存、读写未分配的内存、栈指针读写越界、读写空指针、内存和文件描述符泄漏、调用函数参数错误等。

由于 purify 对内存的分析和记录是在程序运行完成以后才显示，如果需要在程序运行时观测和显示内存状况，以及调试程序就需要调用 Purify 提供的外接 API 函数。

Purify 所支持的操作系统有 Windows、Sun Solaris、HP-UX 和 SGI-IRIX。

5.5　面向对象的软件测试

针对面向对象的开发模型，结合传统的软件测试步骤，可以把面向对象的测试分为面向对象分析的测试、面向对象设计的测试、面向对象编程的测试、面向对象的单元测试、面向对象的集成测试和面向对象的系统测试。

面向对象分析的测试、面向对象设计的测试是对分析结果和设计结果的测试，主要是对分析和设计产生的文本进行测试，是软件开发前期的关键性测试。面向对象编程的测试主要针对编程风格和程序代码实现进行测试，其主要测试内容在面向对象的单元测试和面向对象的集成测试中体现。面向对象单元测试是对程序内部具体单一的功能模块的测试，如果程序是用 C++语言实现，主要就是对类成员函数的测试。面向对象单元测

试是进行面向对象集成测试的基础。面向对象集成测试主要对系统内部的相互服务进行测试，例如，成员函数间的相互作用、类间的消息传递等。

1．面向对象分析的测试

OOA 直接映射问题空间，将问题空间中的实例抽象为对象，用对象的结构反映问题空间的复杂实例和关系，用属性和服务表示实例的特性和行为。对一个系统而言，与传统分析方法产生的结果相反，行为相对稳定，结构则相对不稳定，这更充分反映了现实的特性。OOA 的结果是为 OOD 阶段类的选定和实现、类层次结构的组织和实现提供平台。因此，OOA 对问题空间分析抽象的不完整，最终会影响软件的功能实现，导致软件开发后期大量可避免的修补工作；而一些冗余的对象或结构会影响类的选定、程序的整体结构或增加程序员不必要的工作量。因此，对 OOA 的测试重点应该放在完整性和冗余性方面。OOA 阶段的测试划分为 5 个方面：

（1）对认定的对象的测试。

（2）对认定的结构的测试。

（3）对认定的主题的测试。

（4）对定义的属性和实例关联的测试。

（5）对定义的服务和消息关联的测试。

2．面向对象设计的测试

OOD 采用“造型的观点”，以 OOA 为基础归纳类，并建立类结构或进一步构造成类库，实现分析结果对问题空间的抽象。OOD 归纳的类可以是对象简单的延续，也可以是不同对象的相同或相似的服务。OOD 是 OOA 的进一步细化和更高层的抽象。所以，OOD 与 OOA 的界限通常难以区分。OOD 确定类和类结构不仅能满足当前需求分析的要求，更重要的是通过重新组合或加以适当的补充，能方便实现功能的重用和扩充，以不断适应用户的要求。

因此，对 OOD 的测试，主要针对功能的实现和重用，以及对 OOA 结果的扩展，从如下三方面考虑：对认定的类的测试、对构造的类层次结构的测试、对类库的支持的测试。其中对构造的类层次结构的测试通常基于 OOA 中产生的分类结构的原则来组织，着重体现父类和子类间一般性和特殊性。在当前的问题空间，对类层次结构的主要要求是能在解空间构造实现全部功能的结构框架。为此，测试如下几个方面：

（1）类层次结构是否包含了所有定义的类。

（2）是否能体现 OOA 中所定义的实例关联。

（3）是否能实现 OOA 中所定义的消息关联。

（4）子类是否具有父类没有的新特性。

（5）子类间的共同特性是否完全在父类中得以体现。

3．面向对象编程的测试

典型的面向对象程序具有继承、封装和多态的新特性，这使得传统的测试策略必须

有所改变。封装是对数据的隐藏，外界只能通过被提供的操作来访问或修改数据，这样，降低了数据被任意修改和读写的可能性，降低了传统程序中对数据非法操作的测试。继承使传统测试遇到了一个难题，即对继承的代码究竟如何测试，多态性使得面向对象程序对外呈现出强大的处理能力，但同时却使得程序内“同一”函数的行为复杂化，测试时不得不考虑不同类型具体执行的代码和产生的行为。

面向对象程序是把功能的实现封装在类中，能正确实现功能的类，通过消息传递来协同实现设计要求的功能。正是这种面向对象程序风格，将出现的错误能精确的确定在某一具体的类。因此，在面向对象编程阶段，将测试集中在类功能的实现和相应的面向对象程序风格，主要体现为以下两个方面：

（1）数据成员是否满足数据封装的要求。

（2）类是否实现了要求的功能。

4．面向对象的单元测试

传统的单元测试是针对程序的函数、过程或完成某一定功能的程序模块。沿用单元测试的概念，实际测试类成员函数。传统的单元测试方法在面向对象的单元测试中都可以使用。

面向对象编程的特性使得对成员函数的测试，又不完全等同于传统的函数或过程测试。尤其是继承特性和多态特性，使子类继承或重载的父类成员函数出现了传统测试中未遇见的问题。需要考虑以下问题：

（1）继承的成员函数是否都不需要测试。对父类中已经测试过的成员函数，有两种情况需要在子类中重新测试，第一种是继承的成员函数在子类中做了改动，第二种是成员函数调用了改动过的成员函数的部分。

（2）对父类的测试是否能照搬到子类。只需在父类测试要求和测试用例上添加对子类函数的新的测试要求和增补相应的测试用例。

5．面向对象的集成测试

面向对象的集成测试通常需要在整个程序编译完成后进行。此外，面向对象程序具有动态特性，程序的控制流往往无法确定，因此，也只能对整个编译后的程序做基于黑盒的集成测试。

面向对象的集成测试能够检测出相对独立的单元测试无法检测出的那些类相互作用时才会产生的错误。基于单元测试对成员函数行为正确性的保证，集成测试只关注于系统的结构和内部的相互作用。面向对象的集成测试可以先进行静态测试，再进行动态测试。

静态测试主要针对程序的结构进行，检测程序结构是否符合设计要求，提供一种成为“可逆性工程”的功能，即通过原程序得到类关系图和函数功能调用关系图，将“可逆性工程”得到的结果与 OOD 的结果相比较，检测程序结构和实现是否有缺陷，即通过这种方法检程序设计是否达到了设计要求。

动态测试设计测试用例时，通常需要上述的功能调用结构图、类关系图或者实体关系图为参考，确定不需要被重复测试的部分，从而优化测试用例，减少测试工作量，使得进行的测试能够达到一定的覆盖标准。测试所要达到的覆盖标准可以是达到类所有的服务要求或服务提供的一定覆盖率，依据类间传递的消息，达到对所有执行线程的一定覆盖率，达到类的所有状态的一定覆盖率等。同时，也可以考虑使用现有的一些测试工具来得到程序代码执行的覆盖率。值得注意的是，设计测试用例时，不但要设计确认类功能满足的输入，还应该有意识地设计一些被禁止的例子，确认类是否有不合法的行为产生。例如，发送与类状态不相适应的消息，要求不相适应的服务等。

6．面向对象的系统测试

系统测试应该尽量搭建与用户实际使用环境相同的测试平台，保证被测系统的完整性，对临时没有的系统设备部件，也应有相应的模拟手段。系统测试时，应该参考 OOA 分析的结果，对应描述的对象、属性和各种服务，检测软件是否能够完全“再现”问题空间。系统测试不仅是检测软件的整体行为表现，从另一个侧面看，也是对软件开发设计的再认识。

本章参考文献

[1] 郑人杰. 计算机软件测试技术. 北京：清华大学出版社，2000

[2] 朱鸿，金凌紫. 软件质量保障与测试. 北京：科学出版社，1997

[3] 周涛. 航天型号软件测试. 北京：宇航出版社，1999

[4] 陈文宇. 面向对象的软件测试. 电子科技大学学报，2006（6）：613-617

[5] 刘斌，高小鹏. 嵌入式软件可靠性仿真测试系统研究. 北京航空航天大学学报，2000（8）：490-493

[6] 孙昌爱，靳若明，刘超等. 实时嵌入式软件的测试技术. 小型微型计算机系统，2000（9）：920-924

第 6 章　系统运行和维护

软件正式交付用户以后，即进入漫长的运行维护期。在这一阶段，基本任务是保证软件在这段相当长的时期内能够正常运行。软件维护需要的工作量很大，随着时间的推移，软件维护对于开发商带来的成本压力也越来越大，现在许多软件开发商要把 70%的工作量用在维护已有的软件上，平均来说，大型软件的维护成本高达开发成本的 4 倍左右。为控制维护成本，需要制定长期计划，有效管理系统的演化过程；同时还要制订临时的短期计划，及时处理各阶段的突出问题。

6.1　维护的实施和管理

在系统交付使用后改变系统的任何工作，都被称为维护。与硬件不同，软件系统构建时包含了变化，软件并不会退化或需要周期性的维护。软件的维护活动是基于软件是可维护的这一前提，软件的可维护性考虑应该贯穿于软件开发的各个阶段。

6.1.1　系统可维护性

原则上，任何人造系统，都是可维护的，可见的实体，例如，桌椅板凳、车子、房子、计算机等，只要知道如何组装它们，知道它们的结构，就可以维护。信息系统虽然是不可见的，但是在开发的时候，是非常清楚它们是如何组成的，如何活动的。不管是多么复杂的系统，开发者同样了解它，可以维护它。

依据系统本身的特点，系统可维护性主要由以下三个因素决定：

（1）可理解性。软件系统通常是沿着“子系统—模块—功能—内部处理过程”这样的思路逐步细化的，维护人员通过了解同样的思路，就可以理解整个系统。

（2）可测试性。借助于人工的经验或者先进的测试工具，维护人员可以对运行的系统进行测试，找出问题的所在。

（3）可修改性。任何系统都是通过某种开发工具来完成的，都有源码。因此，从理论上来说，使用同样的工具，维护人员可以对现有系统进行修改、编译，使之重新运行起来。

可以从软件的内部和外部两个方面来度量可维护性。

在软件外部，可以用平均修复时间（Mean Time To Repair，MTTR）来度量软件的可维护性，它指出处理一个有错误的软件需要花费的平均时间。如果用 M 表示可维护指标，那么 $M = 1/(1+MTTR)$。为此，需要详细记录以下信息：分析问题需要的时间、确定

改动方案的时间、执行改动花费的时间、测试改动花费的时间、其他管理浪费的时间。

在软件内部，可以通过度量软件的复杂性来间接度量可维护性。与软件复杂性相关的因素有环路数、软件规模和其他因素。

（1）环路数。这是 McCabe 在 1976 年提出的，因此也称为 McCabe 环路。环路数可以反映源代码结构的复杂度。环路数的计算过程是：分析源码，绘制出等效的控制流图；运用图论的知识计算出控制流图的线性无关路径数，用这个数度量和比较复杂程度。通过观察环路数的增长幅度，可以比较多种维护方案的优劣，选择对环路数影响最小的方案作为最优方案。

（2）软件规模。通常，可以认为软件包含的构件越多，软件就越复杂，可维护性就越差。

（3）其他因素。包括嵌套深度、系统用户数等。

对于软件内部的可维护性，迄今还没有一个突出的、全面的、通用的模型，需要结合不同开发组织自身的经验，建立合适的经验模型。一般来说，可以按如下步骤建立经验模型：首先，确定影响可维护性的若干主要因素，并为其制定尺度；接着，用大量的现有案例，计算出一个包含影响因素及其权值的多项式模型；然后，利用这个经验模型分析新的软件，再根据实际的维护活动的感受进行校准；重复这一校准过程，就能逐步建立起逼近实际的基本稳定的模型。这个模型最常用的是其比较特性，也就是说，可以据此对问题的维护难度进行排序。

6.1.2 维护的分类

系统维护从性质上可以分为纠错型维护、适应型维护、预防型维护和完善型维护。

尽管经过严格的测试，但并不能保证软件中彻底没有了错误，随着运行时间的延续，数据量的积累，各种应用环境的变化，错误仍会顽固的暴露出来，此时就要进行纠错型维护，这种维护主要是为了控制日常的系统功能。

伴随着计算机硬件的新产品、操作系统的新版本的不断推出，或软件运行的基础环境发生了变化，正在运行的软件必须进行适应型维护。

用户逐渐的熟悉软件以后，会提出一些改进需求，为了满足这些需求，就必须进行完善型维护，这样的维护几乎占到维护工作量的一半以上。例如，打印格式的调整、统计口径的增加、业务流程的完善等。

以上三种维护都是用户驱动的，用户是维护需求的提出者，而开发商“为了明天的需要，把今天的方法应用到昨天的系统中”，目的是为了使旧系统焕发新活力，这样的维护就是预防型维护，这种维护所占的比例很小，因为它耗资巨大。

Lientz 和 Swanson 调查发现在 1980 年发现，完善性维护约占 50%，适应性维护约占 25%，纠错性维护约占 21%，其他维护只占 4%。这个调查已是 30 年前的事了，具体的比例数不必深究，现在的情况并没有多大变化。

这里，重点谈一下预防性维护，随着软件技术的发展，软件企业以前用老的软件技术开发的系统的现状大致有三种：

（1）使用情况不理想的，基本停用。

（2）由于用户业务发生了大的变化，原有软件只有部分在用，大部分基本停用。

（3）对于仍能满足大部分用户需求的，大部分仍在使用。例如，某些银行的基础软件等。

但由于开发时间久远，本身的架构和数据结构都不好，文档原来就不全，或者逐渐遗失殆尽。对于这些老系统（主要指第 3 种），显然，不能盲目地尝试去修改程序，没有设计文档的源码犹如天书。而只能通过分析来更多地了解程序结构再去尝试修改，或者在理解原有设计的基础上，运用软件工程方法重写需要变更的部分，或者干脆全部重新设计、编码、测试。其中最后一种方法称为软件再工程，这种维护也就是预防性维护，它的可行性体现在以下几个方面：

（1）可把现有系统做为原型，提高开发生产率。

（2）利用用户多年的使用经验，很容易弄清需求的变更范围。

（3）可以利用逆向和再工程自动化工具，代替人完成大量的重复琐碎的工作。

虽然这种维护需要更大的精力，但一旦成功，效益也是可观的，新系统具有更好的结构，更好的可维护性，它把业务理解的成熟性和技术的先进性结合在一起，会给企业带来丰厚的利润。例如，用 Java 技术重建老系统，将立即获得跨平台的特性，从而扩大了软件的适用范围。

6.1.3　影响维护的因素

系统维护的影响因素很多，主要有以下几个方面：

（1）业务因素。因为系统在线运行，某些系统要保证 7×24 小时运行，维护人员必须寻找一种途径，在不影响用户业务的情况下实现改动。

（2）理解的局限性。统计数据显示，47%的软件维护工作量花在理解要修改的软件上。有时用户的理解也会出现问题，约有超过半数的维护问题源自用户技能或理解的缺乏。同时，维护人员也要具备一些“人际技巧”，努力理解不同用户的思维方式，以说服的方式处理掉一些问题。

（3）对待维护的优先级问题。有时开发商会倾向于维持现有系统的运行，而客户更迫切地需要新功能，甚至一个新系统。

（4）维护人员的积极性。通常，维护人员被认为是第二阶层，程序员大都认为设计和开发比维护工作更具技巧性和挑战性。

（5）测试的困难。测试人员很难预测设计或代码改动带来的影响，使得测试很难做到充分。另外，有些系统只能在测试环境或备份系统中进行测试，上线后则不允许测试，由于无法精确的再现真实环境，也使得测试有一定的局限性。

为尽可能的减低这些因素的影响，维护人员经常要进行权衡，在长期和短期目标之间进行权衡，决定什么时候牺牲质量来换取速度。除此之外，还可以从以下方面来提高软件的可维护性。

1．采用软件工程方法

软件危机从某种意义上可以看成是软件维护的危机，由此诞生的软件工程也可以看成是提高软件可维护性的工程。软件工程的采用，使得软件开发过程进一步规范，强制性的产生了一系列的文档，这些文档从各个阶段、各个方面对软件结构、原理加以说明，极大的丰富了维护所需的资源，增加了软件的可维护性。所以，对于维护人员来说文档比程序源码更为重要。有关软件文档的详细知识和规定，请阅读第 10 章。

2．注重可维护性的开发过程

随着软件的规模不断增大，维护活动耗费的成本急剧攀升。甚至有人预言，如果再不重视软件的可维护性，总有一天，开发商将不得不投入所有的资源进行软件维护，而无力开发新的软件。这虽然有些危言耸听，但在需求分析、设计、编码各阶段加入灵活应变因素的做法，还是得到越来越多的开发者的认可，如此一来，既可以降低维护难度，从而降低维护成本；又可以提高软件的可维护性。

要在开发过程中提高软件的可维护性，必须从如何使软件易于理解、易于测试、易于修改出发进行考虑。具体如下：

（1）在需求分析阶段，应该对将来要改进的和可能会修改的部分加以明确；对软件的跨平台可移植性进行讨论，形成解决方案。

（2）在设计阶段，应该尽量遵循“高内聚低耦合”的模块设计原则，对将来可能要修改的地方，采用灵活的易于扩充的设计方案；考虑跨平台可移植性的设计；加大可重用构件的设计力度；如果各方面条件都满足，应该使用面向对象的设计方法。

（3）在编码阶段，应该采用科学的代码规范，强化注释的力度，保证注释的质量，这一点对于将来的维护非常重要；加大可重用构件的使用力度；同样，如果各方面条件都满足，应该使用面向对象的编码工具。

（4）在测试阶段，一方面，测试的目的本质上是为了减少各种维护的工作量，尤其是纠错型维护。也就是说，测试做好了，以后的维护量就少了；另一方面，测试相关的文档（包括测试大纲、用例设计、测试报告等）是维护后的回归测试的基础，测试阶段要是文档不全，维护后几乎无法进行回归测试，维护的质量也就无法保证，最糟糕的情况莫过于改了旧错，又引入了新错，进入恶性循环。

（5）在维护阶段，要有严格的配置管理，每一次维护工作之后，都要按照配置关联，同步更新维护有关的系统文档（包括维护需求、源代码、设计文档、测试文档）和用户文档（包括用户使用手册等），保证系统的一致性。同时，在大的维护之后，交付之前，一定要及时做好用户的培训工作。否则，就会使用户因为受挫折而产生不满。实际工作中，有些所谓的“错误”可能只是用户使用手册描述不清楚造成的。

6.1.4　维护工作量

维护活动可以分成生产类（例如，确认维护需求、设计、编码、测试、培训等）和非生产类（例如，熟悉原有软件的代码，理解原有软件的结构等）。维护工作量可以用一个模型表示：

$$M = P + K^{c-d}$$

其中，M 是维护用的总工作量，P 是生产类活动的工作量，K 是经验常数，c 是软件的复杂程度，d 是维护人员对软件的熟悉程度。

对于一次具体的维护，确认需求和设计的工作量与问题的难易和大小有关，相对来说比较稳定，编码工作则与软件本身的质量有很大的关系，如果原来的编码格式混乱，注释不清，就会使生产类活动的工作量 P 增大；在软件的复杂度 c 一定的前提下，维护人员对软件的熟悉程度 d 越低，则维护工作量呈指数规律增加；同样，如果由于开发混乱，导致软件复杂度 c 增加，从而使维护人员理解软件的难度增加，对软件的熟悉程度 d 也降低，那么维护工作量会以更快的速度上升。

除了原有软件本身的质量（包括设计质量、代码质量、文档质量和测试质量）对维护工作量有影响外，还有如下一些因素也会影响到维护工作量：

（1）维护工作本身是否规范，是否按软件工程的正确方法进行，对后续的维护工作量的影响同样不可忽视。如果维护工作不规范，代码修改与文档修改不同步，会导致维护后的软件更加复杂，更加难以理解，难以熟悉，维护工作量也会以指数速度增加。

（2）系统的类型不同，维护工作量也有区别。通常，一个系统越依赖于真实世界，就越可能发生变化，也就需要更大的维护工作量。按照对真实世界的依赖程度，软件系统可以分为抽象系统、近似系统和模拟系统。抽象系统描述的问题具有形式化的精确定义，比如涉及标准数值计算方法的计算软件；近似系统是对于真实世界的一个简化的近似方案的描述，例如，围棋软件，虽然围棋的规则是精确的，但由此衍生的走步方案和对弈模拟则几乎是无穷尽的，因此一般的设计都有计算深度的限定；模拟系统则包括广泛的行业应用软件，系统本身就承担着全部或部分业务，是嵌入真实世界中运行的，例如，管理信息系统、ERP 系统、计费系统等。因此，抽象系统的维护工作量最小，模拟系统的维护工作量最大。

（3）硬件因素。不可靠的硬件系统会使软件系统产生一些令人恼火的随机性的问题，使得追踪问题的根源变得更加困难。

上面所讲的都是客观因素，还有维护人员本身的因素，基本上在软件开发队伍中，维护被认为是出力不讨好的工作，维护人员作为软件企业中长期面对用户的角色，经常得承担来自其他开发人员和用户的双重压力，情绪低落，热情不高。同时，部分软件企业对维护不够重视，认为维护是个无底洞，因此在人力和成本的投入上采取敷衍的态度，安排进行维护的人员大多是技术不够扎实的员工，甚至把维护作为培训新员工的渠道，

导致维护人员技术素质一般，又缺乏热情。

希赛教育专家提示：这种局面正在改观，随着软件企业的数量的激增，导致竞争加剧，在某些领域甚至到了白热化的程度，巩固已有的客户，从已有的客户身上寻找新的商机变得越来越重要，维护人员作为服务的窗口，逐渐被重视，因为只有维护做好了，才能巩固已有客户。

6.1.5 维护管理

不管是哪种类型的维护，都需要类似开发的过程，从本质上说，维护过程是修改和压缩了的开发过程。但维护的范围更广，需要跟踪和控制的东西更多。维护的重点是保持对系统日常功能的控制和对系统改动的控制，完善现有可接受的功能，防止系统性能下降到不可接受的程度。

1．建立维护组织

维护工作是需要相当大的投入的，关键行业的企业大都有自己的维护队伍。有实力的软件企业应该建立专业的维护部门，除维护自有产品外，也可承接维护外包业务。对大多数的中小软件企业，虽然没有专门的维护组织，但非正式的明确责任却是非常必要的，需要指定维护负责人，组建临时的维护小组，明确维护流程及各人的职责，可以专设维护配置员，也可由维护负责人兼任。维护小组的工作包括如下活动：

（1）理解系统。

（2）保持系统文档更新。

（3）扩展当前的功能，以容纳新的或变动的需求。

（4）给系统增加新的功能。

（5）找出系统故障或问题的根源。

（6）定位和纠正故障。

（7）回答有关系统运行方式的问题。

（8）重构系统组件。

（9）重新编写系统组件。

（10）删除不再有用的系统组件。

（11）进行系统改动。

如果同时有多个系统需要维护，可再设维护管理员，监督协调各维护小组，结构大致如图 6-1 所示。

对于大型软件系统的维护，设置维护管理员是非常必要的，由其协调和同步各维护小组的工作。维护配置员则对维护活动需要的资源以及维护过程中产生的各种文档进行标识和配置。由于维护几乎必然会带来变动，因此，需要维护配置员至少做好变动控制，包括变动的时间、执行变动的人员、变动的内容、变动的结果、批准变动的人员、变动的通知范围、变动的级别。

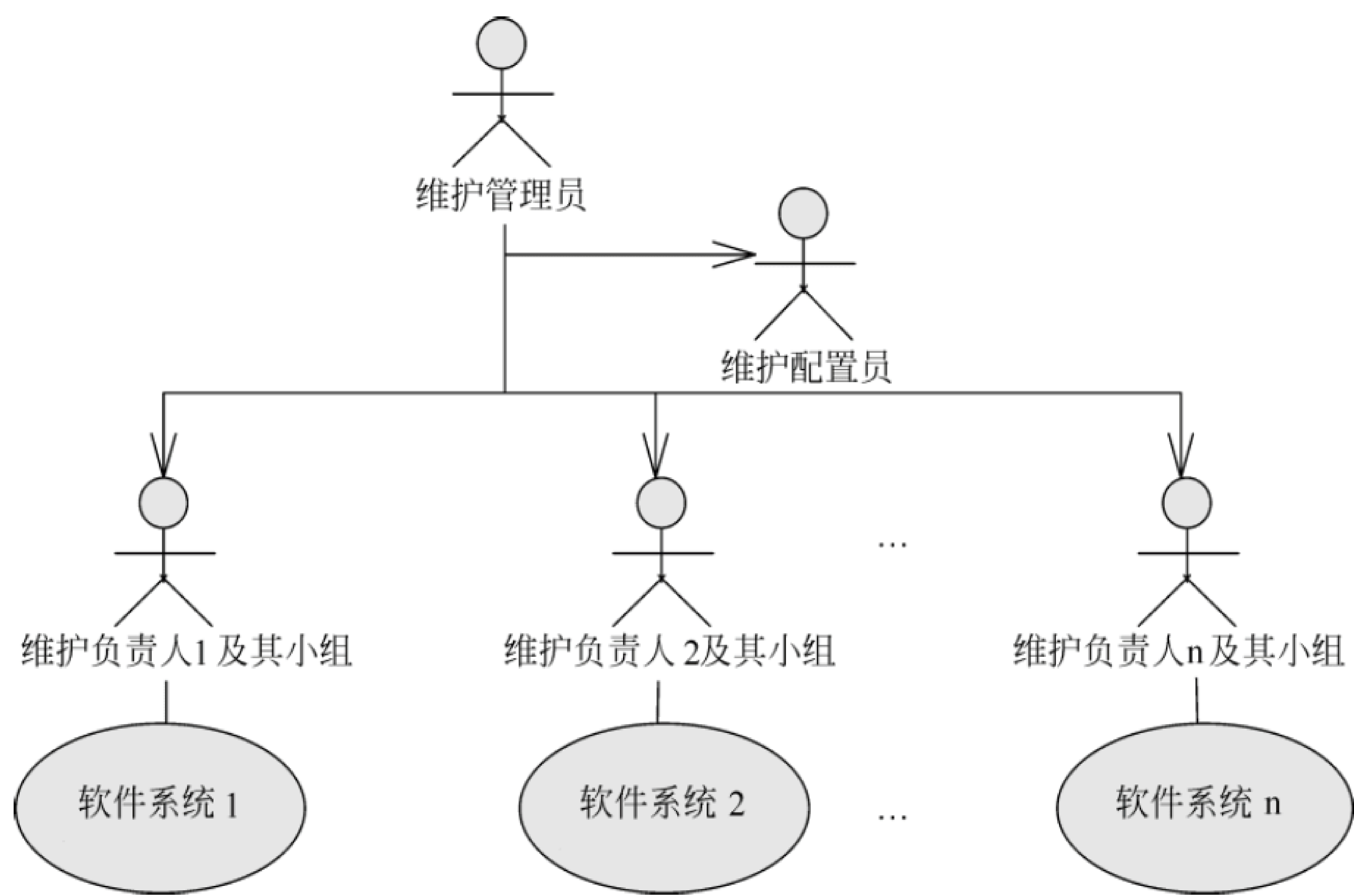


图 6-1　维护的组织结构

2．提出维护需求

维护需求通常由用户提出，维护人员应该给用户提供空白的软件问题报告，并用具体的例子向用户说明如何记录碰到的软件问题，以便规范用户完整描述其维护需求。

软件问题报告通常包括如下内容：

（1）问题发现人姓名及其所在的部门。

（2）发现问题的时间。

（3）发生问题的子系统名称。

（4）问题产生前进行的操作。

（5）问题的界面或描述。

（6）问题的后果描述，比如退出当前操作，退出应用系统还是导致死机。

用户把填好的软件问题报告提交给维护管理员，维护管理员再根据各维护小组的分工，把软件问题报告转给相应的维护小组，由其负责人组织实施维护作业，同时由维护配置员进行维护资源配置管理。

如果同时有多个维护需求，维护管理员则应根据对用户的影响程度，首先确定维护的优先顺序，再依次安排维护活动。

3．实施维护作业

每一次维护活动的实施都要经历确认维护需求、制定维护计划、编码、测试、交付用户等几个步骤。维护作业中各阶段的参与人及其活动如图 6-2 所示。

图 6-2 描述的是一个符合软件工程思想的、规范的维护流程，其中，维护计划包括维护的人员、时间安排、维护需求描述、维护设计及相关测试文档。另外，制定维护风

险控制计划对保证维护按时保质的完成非常重要。

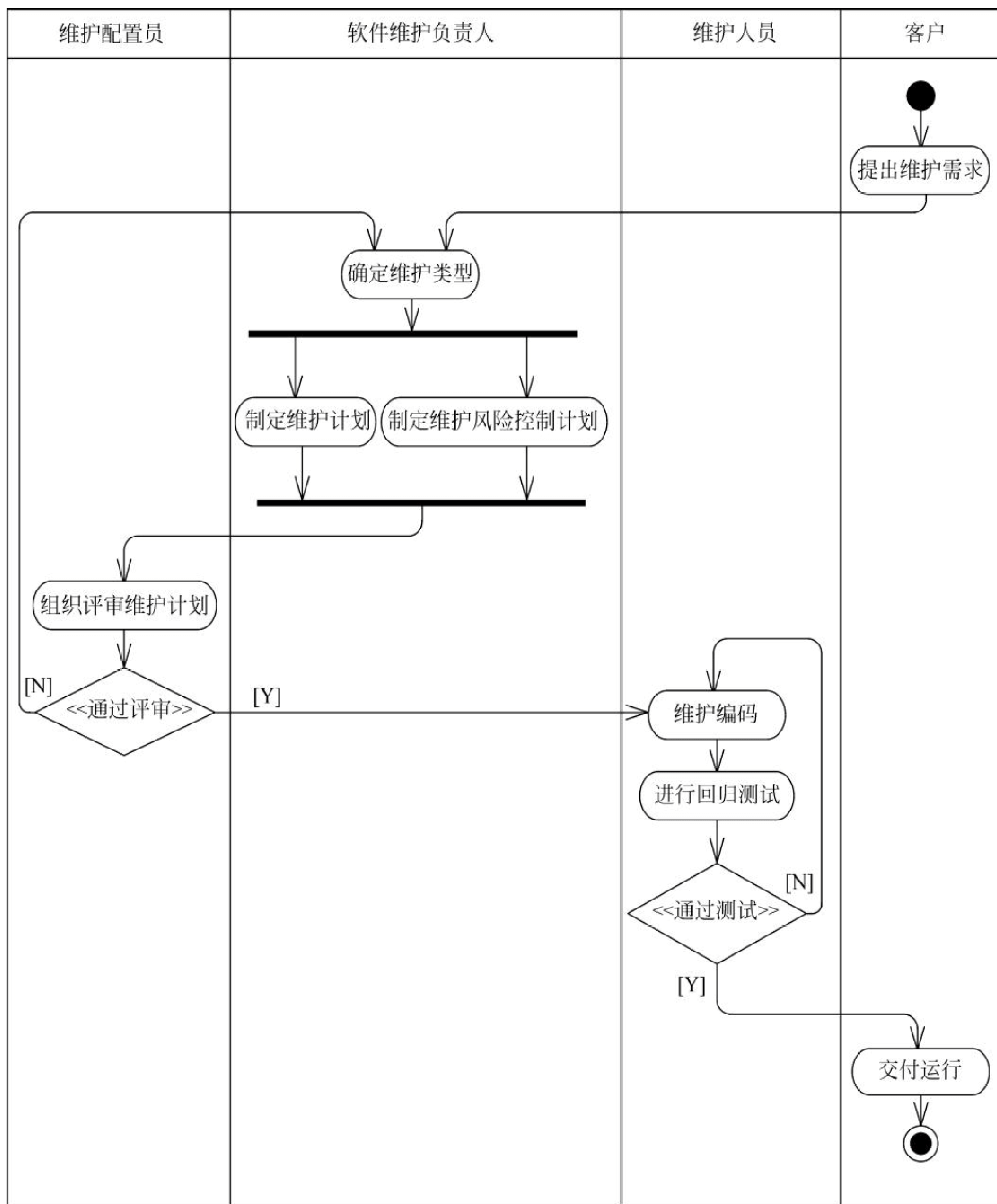


图 6-2 维护作业实施

虽然维护所带来的副作用是客观存在的，不可能完全消除。但这样的工作流程可以大幅降低因管理因素产生的维护副作用(例如，因为配置混乱造成的文档更新不及时等)，保证每一次维护活动都有据可查。

完善的维护设计文档，可以有效降低因修改数据结构带来的副作用，保证与该数据

结构有关的代码段的修改不被遗漏。回归测试则可以有效地查明修改程序代码对原有软件的影响。

希赛教育专家提示：并非所有的维护活动都得完全按照上述流程进行。例如，如果需要维护的软件系统是支撑着客户的核心业务，必须 7×24 小时运行，这样的系统一旦发生恶性软件问题（如数据库崩溃等）时，维护组织就会以“救火队”的方式迅速展开维护，没有时间进行计划评审，此时，软件开发商应派出技术过硬的维护小组，以最大限度的降低风险。

4．记录维护要素

在维护活动中需要及时记录维护的有关信息，用以考察维护技术的有效性，估计软件的“优良”程度，确定维护的实际代价。同时，这些记录将作为后续评价活动的依据。

维护记录的主要内容包括源码行数、使用的开发语言、程序的安装日期、从安装至今程序运行的次数、从安装至今程序失效的次数、程序变动的标识、因变动而增加的源码行数、因变动而删除的源码行数、每个改动耗费的人时数、程序改动的日期、修改者的姓名、维护需求表的标识、新维护需求是否源于以前的维护工作、维护类型、维护开始和完成日期、累计用于维护的人时数、已完成的维护创造的直接和间接的效益。

对于每项维护工作，都要收集上述数据，可以基于这些数据建立起维护数据库。

5．评价维护活动

借助于维护记录，可以对维护工作做一些定量的统计。可以从以下几个方面进行度量：

（1）以前维护工作引起的新维护占总维护需求的比例。

（2）程序的平均失效次数。

（3）每一类维护活动的总人时数及维护工作量占比。

（4）维护需求表的平均周转时间。

（5）维护活动中增加一条源码花费的平均人时数。

（6）维护活动中删除一条源码花费的平均人时数。

（7）按维护的程序、开发语言和维护类型统计程序变动数。

（8）维护分类占比。

度量的结果可以作为以后调整维护工作的参考，对于合理规划维护工作量，优化资源分配，针对性的强化对参与某类维护的人员的技术培训等方面可以起到积极的作用。

6.2　系统的扩展和集成

随着系统的运行和业务的发展，对现有系统进行有效的扩展升级，使其适应当前的应用情况，就成为必然的，也是经济的选择。系统设计时，如果充分考虑了扩展的因素，从系统结构上进行了预留，则同类型业务的扩展则相对容易。

一般地，当客户提出需求变更或增加新的功能时，通常采用的方法首先就是对现有系统进行扩展，以满足这种变化。扩展一般包括延伸型扩展和新建型扩展，前者需要在理解扩展点附件的架构及代码的基础上，以原有方式进行功能扩充；后者则可能会完全另起炉灶，在适应系统整体架构的前提下，增加全新的功能。

通过在基本软件基础上，引入第三方软件包并进行二次开发的方式，可以迅速对系统功能进行扩展。例如，引入具有手写签批功能的控件，可以立即得到支持领导手工批阅公文的功能扩展。但这种引入也是双刃剑，需要在引入前进行充分的研究和分析，确保其能满足目前的扩展需求外，还具有适度前瞻的特性。同时，要求引入详细的设计文档甚至是源码，以保证对引入包的可控性，避免过度依赖第三方技术支持，减低实施风险。

1．系统集成

当组织建成越来越多的应用系统后，系统之间的数据共享和连动运行就成为关注的焦点。此时，就需要采用某种方法把多个应用系统串接在一起协同工作，这就叫系统集成。根据集成的程度，可以把系统集成分成界面集成、数据集成、业务集成三种。

（1）界面集成（表示集成）：把多个系统放到同一界面，便于访问和切换。

（2）数据集成：系统间通过访问对方数据库或向对方开放数据接口，使得各系统处理的数据可以达到共享。

（3）业务集成（功能集成）：各系统的单个流程被集成为一个更广泛的大流程，使得业务得以延展和贯通。

不同类型的系统集成，需采用不同的方法。界面集成相对简单，仅需增加一个统一的界面，把各程序的入口集中起来即可，无须了解各系统细节；数据集成则需要了解各系统的数据结构，通过直接读取对方数据的方式实现数据共享，这种方法需要的工作量比前者大；业务集成是最复杂的集成方式，除要了解数据结构外，还需进行业务衔接，工作量最大，通常采用 Web Service 方式，封装服务接口供其他业务调用。

有关系统集成更多的知识，请阅读本套丛书中的《系统分析师技术指南（2009 版）》的第 9 章。

2．系统扩展

系统扩展和集成分别属于深度维护和广度维护活动，和开发过程类似，都需要经过需求分析、设计、编码、测试和实施的完整流程。需要分析人员、设计人员、编码人员、测试人员和实施人员的参与，需要过程管理人员进行项目管理，系统集成还特别需要组织的高层人员参与协调与沟通。

系统扩展的重点在设计阶段，为达到平滑扩展，需要仔细研究扩展点附近的软件环境，要避免因扩展引起原有系统的动荡，要进行细致的回归测试。

系统集成的重点在分析阶段，为达到无缝集成，需要仔细分析业务，尤其是业务关联点。避免过度耦合、深度介入，增加集成复杂度。另外，还需组织的高层领导强势协

调，强调全局观念，互相配合，方能顺利进行集成。

系统扩展和集成的测试要进行全面的回归，不能改头测头、改脚测脚，系统集成尤其要重视接口的测试、流程流畅性的测试。

6.3　新旧系统的转换交接

当新系统似乎开发完毕，要取代原来的系统时，系统过渡就是设计师不得不面对的问题。不幸的是，这个问题比许多人想象的要复杂，和软件开发一样，存在着许多冲突和限制。例如，费用、客户关系、后勤保证和风险等。设计师需要考虑的问题也很多，其中比较重要的几个问题是：

（1）如果同时运行两个系统，会给客户造成多大的开销。

（2）如果直接运行新系统，客户面对的风险有多大。

（3）对新系统试运行时的查错和纠错，以及出现严重错误而导致停止运行时的应急措施。

（4）客户运行新系统将面临的不利因素有哪些。

（5）人员的培训。

6.3.1　新旧系统的转换策略

根据新旧系统比较的结果，可以得到未来新系统建设完成后的转换策略。通常新旧系统的转换有三种主要的策略。

1．直接转换策略

新系统是完全重构的系统，可能采用了全新的技术平台和软件来构建，或者用户业务和使用方式发生了剧烈变化，对原有系统只能进行抛弃处理。采用这种策略的优点是新系统能够非常灵活地适应业务需要，功能齐全、结构合理、系统稳定、扩展性强，整个信息系统的利用率比较高。但也存在着一些问题，主要是：新旧系统之间的转换代价比较大；由于需要一套比较完整的业务需求，开发新系统的周期比较长，一次性投资巨大，未经广泛使用并证明是成熟可靠的新技术平台通常具有一定的技术风险；另外旧系统通常积累下了大量的业务数据、必须将业务数据的录入、转换、检查以及在新系统中的重建作为重要的工作进行考虑，尽量减小在新旧系统转换的时候对用户现有业务的冲击；最后，考虑新旧系统转换还需要考虑诸如：维持新系统运行的日常开销、由于使用习惯改变带来的学习时间、培训人员的成本等因素。

2．逐步转换策略

这种策略是部分继承原有系统，部分进行新系统的更新和开发的技术路线，每成熟一部分新系统软件，就更新一部分旧系统软件，最终采取渐进的方式，过渡到新的软件平台上来。这种策略的优点是，新旧系统的转换震动比较小，用户容易接受，但也由于

是采用渐进式，导致新旧系统的转换周期加大，同时由于需求的变化，给新系统的稳定造成比较大的影响。在转换过程中，需要开发新旧系统之间的接口，还需要制订阶段性的转换目标和计划。

3．并行转换策略

这种策略是在一定的阶段并行使用新旧系统。将同样的业务在新旧系统中都完成一次。然后在确认新系统已经可以稳定工作后、停止旧的系统和全面启用新系统。并行转换策略会较多增加用户的工作量，并且难以控制新旧系统中的数据变化，因此很少使用。

通常，对于现有信息系统比较稳定、能够适应自身业务发展需要的建设单位，或新旧系统转换风险很大（例如，订票系统或银行的中间业务系统），可以采用渐进方式；对于现有信息系统本身就存在问题，例如，已经不能满足业务需要，存在安全、性能等方面问题的，应采用直接转换策略。

新旧系统比较的结果，可提供新旧系统转换策略选择上的基本建议。如果逐步转换策略可用，则可以由客户根据自身的具体情况决定采用哪种策略；如果新旧系统差异很大，无法选择逐步转换策略，通常要在直接转换前，对新系统进行多次测试、排错、试运行和评估。

6.3.2 软件再工程

再工程是演化中常用的可行技术之一，通过再工程演化遗留系统时，可以识别出遗留系统中的可复用成分，然后对其进行再工程，以提取出可复用构件。对于已有的可复用构件，软件再工程也可以对其进行适应性修改和维护，以进一步提高其复用率，延长其生命周期。

软件再工程在软件开发中应用得比较多，例如，单机版的遗留系统改造为网络版、C/S 的遗留系统改造为 B/S、非结构化的遗留系统改造为结构化、构件化改造等软件行为，这些行为有的是全部再生，有的是局部再生。对于很久以前开发的软件，由于未采用软件工程思想等各种原因，导致文档缺失，甚至只剩下能运行的软件系统，这种情况大多考虑进行局部再生；对于近期采用软件工程思想开发的软件，由于文档相对齐全，则可以进行全部再生。

通常，软件的再生工程包括筛选分析、逆向工程和补充开发三个阶段。

1．筛选分析

这是局部再生工程中首先要进行的活动，通过对遗留系统的详细审查和分析，选定现有的软件系统中需要进行再生的模块，通常应该着重对以下三类模块进行考察：

（1）确定将使用多年的。

（2）正在成功运行的。

（3）近期将做重大变更的。

通过筛选和分析，也为对遗留系统是采用维护、再工程还是新开发提供参考。

2．逆向工程

软件的逆向工程就是分析一个程序的过程，最大程度地建立比源码更抽象的高级表达，它也是一个恢复设计结果的过程。逆向工程工具可以从现有的软件代码中抽取有关的数据、体系结构和处理过程的设计信息。

Scott R. Tilley 将逆向工程定义为包含三个步骤的过程。

（1）建模：即构造应用程序的领域模型。

（2）抽取：利用抽取机制从目标系统中收集原始数据。

（3）抽象：对目标系统进行抽象并能清晰地表示出来，以便更准确的理解系统结构。

逆向工程主要包括以下活动：

（1）文档重构。利用现有的源码，逆向生成系统文档。对于遗留系统，这项活动非常耗费精力，应分轻重缓急分别对待。对于相对稳定的程序，暂且使其保持现状；对于当前正在修改的部分建立完整的文档，其他部分则在使用中逐步建立文档；对于支撑用户核心业务的程序，必须建立完整的文档，但也最好想方设法的降低建立文档的工作量。

（2）数据重构。指重构数据结构。这是一项全范围的活动。首先进行逆向工程，分解出当前的数据结构，再进行重构。数据结构是软件的基础信息，对其进行重构必然会导致代码重构。

（3）代码重构。这是最常见的再生工程活动。首先用重构工具分析源码，标出非结构化的部分，然后自动重构问题代码，经过复审和测试生成最终的重构代码，同时更新有关文档。

3．补充开发

通过上面几个活动，总会遗留一些工作必须进行开发。新开发必须应用软件工程的原理、概念、技术和方法来进行，以方便日后二次再工程。

6.3.3　数据转换和迁移

数据转换和迁移是新旧系统转换交接的主要工作之一。为使数据能平滑迁移到新系统中，在新系统设计阶段就要尽量保留旧系统中合理的数据结构，这样才能尽可能降低数据迁移的工作量和难度。但是，由于新系统的引入，数据迁移工作是个必然的过程，旧系统中的数据可以通过定制开发的转换工具软件翻译为新系统可以接受的数据格式。

数据转换和迁移工作的原则是数据不丢失。许多无法自动转换的数据，必要时通过手工方式补录进入新系统。数据转换和迁移的流程一般如下：

（1）规划转换方案。对旧系统的数据进行分类统计，分别制定转换思路和方法，并对风险进行充分评估，给出避险预案。

（2）测试转换方案。按照方案中给出的方法进行测试，以验证是否能进行正确转换，要尽量模拟新旧系统环境，使得测试更接近实际。

（3）实施转换方案。经过充分测试后，即可进入现场实施转换操作。备份旧系统中

的数据；通过数据迁移程序自动执行转换及迁移工作，或通过命令编辑器导入升级 SQL 语句，手工执行命令，完成关系型数据库的迁移；再完成其他类型数据的迁移；若数据转换迁移工作部分或全部失败，执行应急恢复预案，把备份的旧数据还原到旧系统中。

（4）验收转换方案。对转换后的数据进行验证，以保证新系统正常上线投运。

数据转换迁移工作中还要特别注意的是转换、迁移工作的周期，必须在用户许可的时间段内完成，否则要把系统恢复到转换前的状态。

6.4 系统日常运行管理

系统的日常运行管理包括实时网络监控、应用运行保障、故障管理和资产管理等服务。

1．组织机构及职责

为有效开展运行管理工作，一般需要设置：客服部、系统部、安全部、网络部、机房部、应用产品部、质量管理部、配件库、专家组等部门。工作职责如下：

（1）贯彻执行各项管理制度，组织落实各项工作措施和管理考核工作，完成各项运行维护工作任务。

（2）负责业务故障的申告受理。

（3）负责网络及设备远程监控操作和支撑工作，主动发现网络、设备和业务应用运行过程中的故障或隐患，进行预处理、派单、时限管控，是相应客户。

（4）负责对故障处理情况进行统计分析和服务质量管控。

（5）负责对网络的日常运行、故障处理进行质量分析并提出整改意见，编写和提交故障处理报告。

（6）负责按服务规范完成所承接的客户端施工作业任务。

（7）按照服务约定开展网络及设备日常巡检、例行测试等维护作业。

（8）负责资料及维护信息的整理、保管、保密。

（9）负责现场服务工作量及质量管控的相关统计、分析工作。

（10）负责合作方现场派驻人员的培训、选派和日常管理，向管理考核部门提交考核意见。

2．工作基本制度

根据运维工作的需要，运维部门可以设置客服经理岗、座席人员岗、维护管理岗（含系统经理岗、安全经理岗、网络经理岗、机房经理岗、应用产品经理岗、质量管理经理岗），质量分析岗，一线技术支持岗、二线技术支持岗、三级技术支持岗等，并且需要规定每个岗位的职责。

同时，要建立健全的监控值班和交接班制度，监控值班包括远程监控和现场监控两类；建立健全备份管理制度、日常安全管理制度、配置管理制度，以及变更、升级、割

接管理制度。

3．日常的故障对策与恢复

（1）系统操作故障。在日常操作中，由于用户的不会使用、操作失误、非正常操作等引发系统不能正常使用，此类故障多由于培训力度不够造成，解决办法只能是加强培训和个别指导。

（2）系统服务故障。在运行过程中出现的常见问题，例如，运行持久性、运行不稳定性、短期内无法处理或集成第三方产品问题而无法解决等，此类故障一方面要有应急优化方案，例如，定期重启服务等；另一方面，要进行深入分析，争取从源头予以解决。

（3）系统灾难恢复。当遇不可抗外力，例如，非计划停电，关键设备硬件失灵等致系统完全瘫痪，无法对外提供服务。此类故障通常会借助日常的数据备份、应用备份等信息，尽快使系统恢复运行，当然会损失部分数据，只要处理及时，其影响一般客户都能理解和接受。

6.5　系统服务质量评价

通过建立指标体系，可以方便地衡量系统的使用效率；通过对指标的监测，可以及时的对系统使用效率进行跟踪。

1．质量指标

（1）网络质量指标。包括：链路可用率、点到点(端到端)的时延、端到端的丢包率、主要链路带宽利用率、设备/系统可用率、服务器 CPU 峰值利用率、服务器 CPU 平均利用率、服务器内存峰值利用率、网管系统的可用率、重大通信阻断发生次数等。

（2）服务质量指标。故障修复及时率，在指定的统计期内，在规定时限内修复的故障数与监控系统营业系统发生故障的总次数之比。

（3）网络能力指标。包括：设备端口占用率、出口带宽峰值占用率等。

（4）安全质量指标。包括：安全漏洞及时修复率（对安全漏洞进行扫描）、高风险重要设备存在率（针对高风险安全漏洞进行扫描）、重大安全事件发生频度、重大网络安全事件。其中重大网络安全事件包括：

① 由网络安全问题引起的重要支撑系统、各种业务系统服务中断 30 分钟以上。

② Web 网站主页被非法修改。

③ 由病毒或其他恶意代码引起的局部网络瘫痪 30 分钟以上。

④ 重要设备（包括主机和网络设备）被攻击者侵入并获得访问控制权限。

⑤ 由于安全设置不当导致系统被攻击者利用而对其他网络实施攻击。

2．质量检查

质量检查可采取定期检查、定期抽查、不定期检查、专项检查和专项抽查等方式。质量检查的内容包括：

（1）各类设备维护保养情况，重点设备的完好率、可靠性、稳定性。

（2）各项维护作业计划的执行落实情况。

（3）配置项信息建档的及时性、完整性和规范性、维护更新的及时性和准确性。

（4）运行现场（含客户端）的管理。

（5）质量分析所提整改、优化措施落实情况及效果。

（6）工作流程执行情况，以及重大故障处理分析情况、应急预案和服务预案制订、执行情况。

3．质量统计与分析

质量统计与分析的内容如下：

（1）网络、设备的主要运行参数及变化情况。

（2）机房的主要环境参数变化情况。

（3）故障申告、处理情况。

（4）各类网络、设备的参数变更情况。

（5）各类应用系统的版本、功能变更情况。

遇到下列情况，应立即召开专题分析会，研究改进措施；如经分析具有普遍性或影响重大，应立即召开专题分析会，研究和制定改进措施。

（1）发生重大系统故障或核心设备故障。

（2）网络、设备在使用过程中发生重复性故障。

（3）网络、设备在使用过程中发生原因不明的故障，且影响业务使用。

（4）公共性、基础性应用平台、应用软件，发生重复性、功能性故障。

（5）当出现重大故障或服务问题时，运行管理部门应立即组织调查，及时召开专题分析会进行分析，制定解决办法或整改措施。合同中有约定的，还应定期与客户召开质量分析联席会议，通报质量统计分析情况，研究解决存在的问题。

运行管理部门应定期根据检查结果，如实填写应用系统及支撑设备的质量统计报表。

本章参考文献

[1] 张海藩. 软件工程导论. 第 4 版. 北京：清华大学出版社，2003

[2] 吴丹，史争印，唐忆. 软件工程理论与实践. 第 2 版. 北京：清华大学出版社，2003

第 7 章　系统可靠性分析与设计

系统的可靠性分析与设计是历年必考的知识点之一，一般上午有一到两道试题，内容主要为系统的故障模型、系统的可靠性模型、组合模型的可靠性计算、马尔柯夫模型的可靠性计算以及三模冗余、N 模冗余和信息校验码等方面；另外下午试题一也经常考察软件可靠性分析与计算、系统可靠性评估等概念与理论的实际工程运用等内容。因此大家应重点复习掌握相关的知识。

7.1　可靠性概述

与可靠性相关的概念主要有：可靠度、可用度、可维度、平均无故障时间、平均故障修复时间以及平均故障间隔时间等。

（1）可靠度。系统的可靠度 $R(t)$是指在 t=0 时系统正常的条件下，系统在时间区间[0，t]内能正常运行的概率。

（2）可用度。系统的可用度 $A(t)$是指系统在时刻 t 可运行的概率。

（3）可维度。系统的可维度 $M(t)$是指系统失效后，在时间间隔内被修复的概率。

（4）平均无故障时间。可靠度为 $R(t)$的系统的平均无故障时间（Mean Time To Failure，MTTF）定义为从 t=0 时到故障发生时系统的持续运行时间的期望值：

$$\text{MTTF} = \int_0^{\infty} R(t)\mathrm{d}t$$

如果 $R(t) = e^{-\lambda t}$ 则：

$$\text{MTTF} = 1/\lambda$$

λ 为失效率，是指器件或系统在单位时间内发生失效的预期次数，在此处假设为常数。

（5）平均故障修复时间。可用度为 $A(t)$的系统的平均故障修复时间（Mean Time To Repair，MTTR）可以用类似于求 MTTF 的方法求得。

设 $A_1(t)$是在风险函数 $Z(t)$=0 且系统的初始状态为 1 状态的条件下 $A(t)$的特殊情况，则：

$$\text{MTTR} = \int_0^{\infty} A_1(t)\mathrm{d}t$$

此处假设修复率 $\mu(t) = \mu$（常数），修复率是指单位时间内可修复系统的平均次数，则：

$$\text{MTTR} = 1/\mu$$

（6）平均故障间隔时间。平均故障间隔时间（Mean Time Between Failure，MTBF）常常与 MTTF 发生混淆。因为两次故障（失败）之间必然有修复行为，因此，MTBF 中应包含 MTTR。对于可靠度服从指数分布的系统，从任一时刻 t_0 到达故障的期望时间都是相等的，因此有：

$$\text{MTBF} = \text{MTTR} + \text{MTTF}$$

7.2　故障模型和可靠性模型

故障直接导致系统功能失效，了解系统的故障模型是进行可靠性分析和评估的基础。因此，本节首先介绍故障模型，然后再介绍可靠性模型。

7.2.1　故障模型

一个故障可能由物理失效、不适当的系统设计、环境影响或系统的操作员失误所引起。

1．故障的来源及表现

首先介绍几个概念。

失效：硬件的物理改变。

故障：由于部件的失效、环境的物理干扰、操作错误或不正确的设计引起的硬件或软件中的错误状态。

错误（差错）：故障在程序或数据结构中的具体位置。错误与故障位置之间可能出现一定距离。故障或错误有如下几种表现形式。

- 永久性：描述连续稳定的失效、故障或错误。在硬件中，永久性失效反映了不可恢复的物理改变。
- 间歇性：描述那些由于不稳定的硬件或变化着的硬件或软件状态所引起的、仅仅是偶然出现的故障或错误。
- 瞬时性：描述那些由于暂时的环境条件而引起的故障或错误。

永久性失效会导致永久性故障。间歇性故障可能由不稳定、临界稳定或不正确的设计所引起。环境条件会造成瞬时性故障。所有这些故障都可能引起错误。不正确的设计和操作员失误会直接引起错误。由硬件的物理条件、不正确的硬件或软件设计，或不稳定但重复出现的环境条件所引起的故障可能是可检测的，并且可以通过替换或重新设计来修复；然而，由于暂时的环境条件所引起的故障是不能修复的，因为其硬件本身实际上并没有损坏。瞬时和间歇故障已经成为系统中的一个主要错误源。

2．常用的故障模型

故障的表现形式千差万别，为此利用所谓故障模型对千差万别的故障表现进行抽象。故障模型可以在系统的各个级别上建立。一般说来，故障模型建立的级别越低，进

行故障处理的代价也越低，但故障模型覆盖的故障也越少。如果在某一级别的故障模型不能包含故障的某些表现，则可以用更高一级别的模型来概括。

（1）逻辑级的故障模型。包括固定型故障、短路故障、元件的开路故障和桥接故障。固定型故障指电路中元器件的输入或输出等线的逻辑固定为 0 或固定为 1。例如，某线接地、电源短路或元件失效等都可能造成固定型故障；短路故障指一个元件的输出线的逻辑值恒等于输入线的逻辑值；元件的开路故障是元件的输出线悬空，逻辑值可根据具体电路来决定；桥接故障指两条不应相连的线连接在一起而发生的故障，也称为线间桥接故障。

（2）数据结构级的故障。故障在数据结构上的表现称为差错。常见的差错有独立差错、算术差错和单向差错。独立差错是指一个故障的影响表现为使一个二进制位发生改变；算术差错是指一个故障的影响表现为使一个数据的值增加或减少 2^i (i=0，1，2，…)；单向差错是指一个故障的影响表现为使一个二进制向量中的某些位朝一个方向（0 或 1）改变。

（3）软件故障和软件差错。软件故障是指软件设计过程造成的与设计说明的不一致，软件故障在数据结构或程序输出中的表现称为软件差错。与硬件不同，软件不会因为环境应力而疲劳，也不会因为时间的推移而衰老。因此，软件故障只与设计有关。常见的软件差错有以下几种。

- 非法转移：程序执行了说明中不存在的转移。
- 误转移：程序执行了尽管说明中存在，但依据当前控制数据不应进行的转移。
- 死循环：程序执行时间超过了规定界限。
- 空间溢出：程序使用的空间超过了规定的界限。
- 数据执行：指令计数器指向数据单元。
- 无理数据：程序输出的数据不合理。

（4）系统级的故障模型。故障在系统级上的表现为功能错误，即系统输出与系统设计说明的不一致。如果系统输出无故障保护机构，则故障在系统级上的表现就会造成系统失效。

7.2.2　可靠性模型

常用的系统可靠性模型有时间模型、故障植入模型和数据模型。

1．时间模型

时间模型中最著名的是由 Shooman 提出的可靠性增长模型，这个模型基于这样一个假设：一个软件中的故障数目在 t=0 时是常数，随着故障被纠正，故障数目逐渐减少。

在此假设下，一个软件经过一段时间的调试后剩余故障的数目可用下式来估计：

$$E_r(\tau)=E_0/I-E_c(\tau)$$

其中，τ 为调试时间，$E_r(\tau)$ 为在时刻 τ 软件中剩余的故障数，E_0 为 $\tau=0$ 时软件中

的故障数，$E_c(\tau)$为在[0，τ]内纠正的故障数，I为软件中的指令数。

由故障数$E_r(\tau)$可以得出软件的风险函数：

$$Z(t) = C \bullet E_r(\tau)$$

其中C是比例常数。

于是，软件的可靠度为：

$$R(t) = e^{-\int_0^t z(t)\mathrm{d}t} = e^{-c(E_0/I - E_c(\tau))}$$

软件的平均无故障时间为：

$$\mathrm{MTBF} = \int_0^\infty R(t)\mathrm{d}t = \frac{1}{C(E_0/I - E_c(\tau))}$$

在 Shooman 的模型中，需要确定在调试前软件中的故障数目，这往往是一件很困难的任务。

2．故障植入模型

故障植入模型是一个面向错误数的数学模型，其目的是以程序的错误数作为衡量可靠性的标准，模型的原型是由 Mills 在 1972 年提出的。Mills 提出的故障植入模型的基本假设如下：

（1）程序中的固有错误数是一个未知的常数。

（2）程序中的人为错误数是按均匀分布随机植入。

（3）程序中的固有错误数和人为错误被检测到的概率相同。

（4）检测到的错误被立即改正。

用N_0表示固有错误数，N_1表示植入的人为错误数，n表示检测到的错误数，ξ表示被检测到的错误中的人为错误数，则

$$P_r\{\xi = y, N_0, N_1, n\} = \frac{\begin{pmatrix} N_1 \\ y \end{pmatrix}\begin{pmatrix} N_0 \\ n-y \end{pmatrix}}{\begin{pmatrix} N_1 + N_0 \\ n \end{pmatrix}}$$

对于给定的N_1和n，在测试中检测到的人为错误数为k，用最大似然法求解可得，固有错误数N_0的点估计值为：

$$\hat{N}_0 = \left.\frac{N_1(n-k)}{k}\right|_{\text{integer}}$$

考虑到实施植入错误时遇到的困难，Basin 在 1974 年提出了两步查错法，这个方法是由两个错误检测人员独立对程序进行测试，检测到的错误立即被改正。用N_0表示程序中的固有错误数，N_1表示第一个检测员检测到的错误数，n表示第二个检测员检测到的错误数，用随机变数η表示两个检测员检测到的相同的错误数，则：

$$P_r\{\eta = y, N_0, N_1, n\} = \frac{\binom{N_1}{y}\binom{N_0 - N_1}{n - y}}{\binom{N_0}{n}}$$

如果实际测得的相同错误数为 k，则程序固有错误数 N_0 的点估计值为：

$$\hat{N}_0 = \left.\frac{N_1 n}{k}\right|_{\text{integer}}$$

3．数据模型

在数据模型下，对于一个预先确定的输入环境，软件的可靠度定义为在 n 次连续运行中软件完成指定任务的概率。最早的一个数据模型是 Nelson 于 1973 年提出的，其基本方法如下。

设说明所规定的功能为 F，程序实现的功能为 F'，预先确定的输入集：

$$E = \{e_i : i = 1, 2, \cdots, n\}$$

令导致软件差错的所有输入的集合为 E_e，即：

$$E_e = \{e_j : e_j \in E \ \text{and} \ F'(e_j) \neq F(e_j)\}$$

则软件运行一次出现差错概率为：

$$P_1 = |E_e| / |E|$$

一次运行正常的概率为：

$$R_1 = 1 - P_1 = 1 - |E_e| / |E|$$

在上述讨论中，假设了所有输入出现的概率相等，如果不相等，且 e_i 出现的概率为：

$$p_i = (i = 1, 2, \cdots, n)$$

则软件运行一次出现差错的概率为：

$$p_1 = \sum_{i=1}^{n} (Y_i \bullet p_i)$$

其中：

$$Y_i = \begin{cases} 0 & \text{如果} \ F'(e_i) = F(e_i) \\ 1 & \text{如果} \ F'(e_i) \neq F(e_i) \end{cases}$$

于是，软件的可靠度（n 次运行不出现差错的概率）为：

$$R(n) = R_1^n = (1 - P_1)^n$$

只要知道了每次运行的时间，上述数据模型中的 $R(n)$就很容易转换成时间模型中的 $R(t)$。

7.3　可靠性分析和可靠度计算

组合模型和马尔可夫模型是计算容错系统可靠性最常用的方法。

7.3.1 组合模型

一个系统只要满足以下条件，就可以用组合模型来计算其可靠性。

（1）系统只有两种状态：运行状态和失效状态。

（2）系统可以划分成若干个不重叠的部件，每个部件也只有两种状态：运行状态和失效状态。

（3）部件的失效是独立的。

（4）系统失效当且仅当系统中的剩余资源不满足系统运行的最低资源要求（系统的状态只依赖于部件的状态）时。

（5）已知每个部件的可靠性，可靠性指可用度或可靠度等概率参数。

组合模型的目标就是根据各部件的可靠性 R_i 来计算系统的可靠度 R_{sys}，组合模型的基本思想如下。

1．枚举所有系统状态

假设系统被划分为 n 个部件，则系统状态是一个 n 维向量，$q=(S_1, S_2, \cdots, S_n)$，其中：

S_i={0，如果部件 i 处于运行状态；1，如果部件 i 处于失效状态（i=1, 2, …, n）}，一个具有 n 个部件的系统共有 2^n 个状态。

2．计算每个系统状态的概率

系统状态的概率是指系统处于该状态的概率。设系统状态 $q=(S_1,S_2,\cdots,S_n)$，q 的所有 0 分量对应的部件用 A_0 来表示（A_0 是所有处于运行状态的部件的集合），q 的所有 1 分量对应的部件用 A_1 来表示（A_1 是所有处于失效状态的部件的集合）。于是，系统状态取得概率为：

$$P_q=(\prod_{i\in A_0}R_i)(\prod_{j\in A_1}(1-R_j))$$

7.3.2 可靠性计算

以理想情况下的串联系统、并联系统、并串联系统和串并联系统为例加以阐述。

1．串联系统

在一个由 n 个模块（部件）构成的系统中，如果系统中任意一个模块失效将导致系统失败（系统的最低资源要求是所有模块全部运行，只有全 0 系统（0，0，…，0）能够使系统运行）。这种系统可抽象地看成是一个如图 7-1 所示的串联系统。

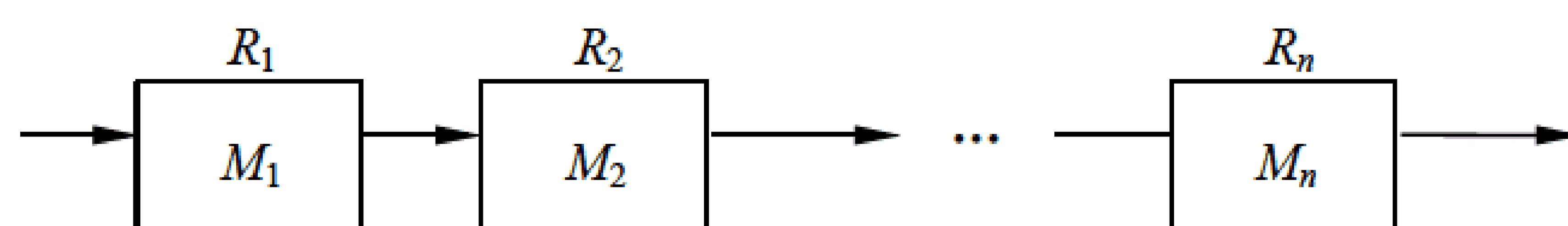


图 7-1 串联系统

假设 R_i 是模块 M_i 的可靠度，则串联系统的可靠度为：

$$R_{sys} = \prod_{i=1}^{n} R_i$$

串联系统的失效率为：

$$Q_{sys} = 1 - R_{sys} = 1 - \prod_{i=1}^{n} R_i = 1 - \prod_{i=1}^{n}(1 - Q_i)$$

其中，Q_i=1–R_i 是模块 M_i 的失效概率。

2．并联系统

在一个由 n 个模块（部件）构成的系统中，如果只要有一个模块可运行，系统就可运行（系统的基本资源是一个模块，除了全 1 系统状态（1，1，1，…，1）外，系统都是可运行的）。这种系统可抽象地看成是一个如图 7-2 所示的并联系统。

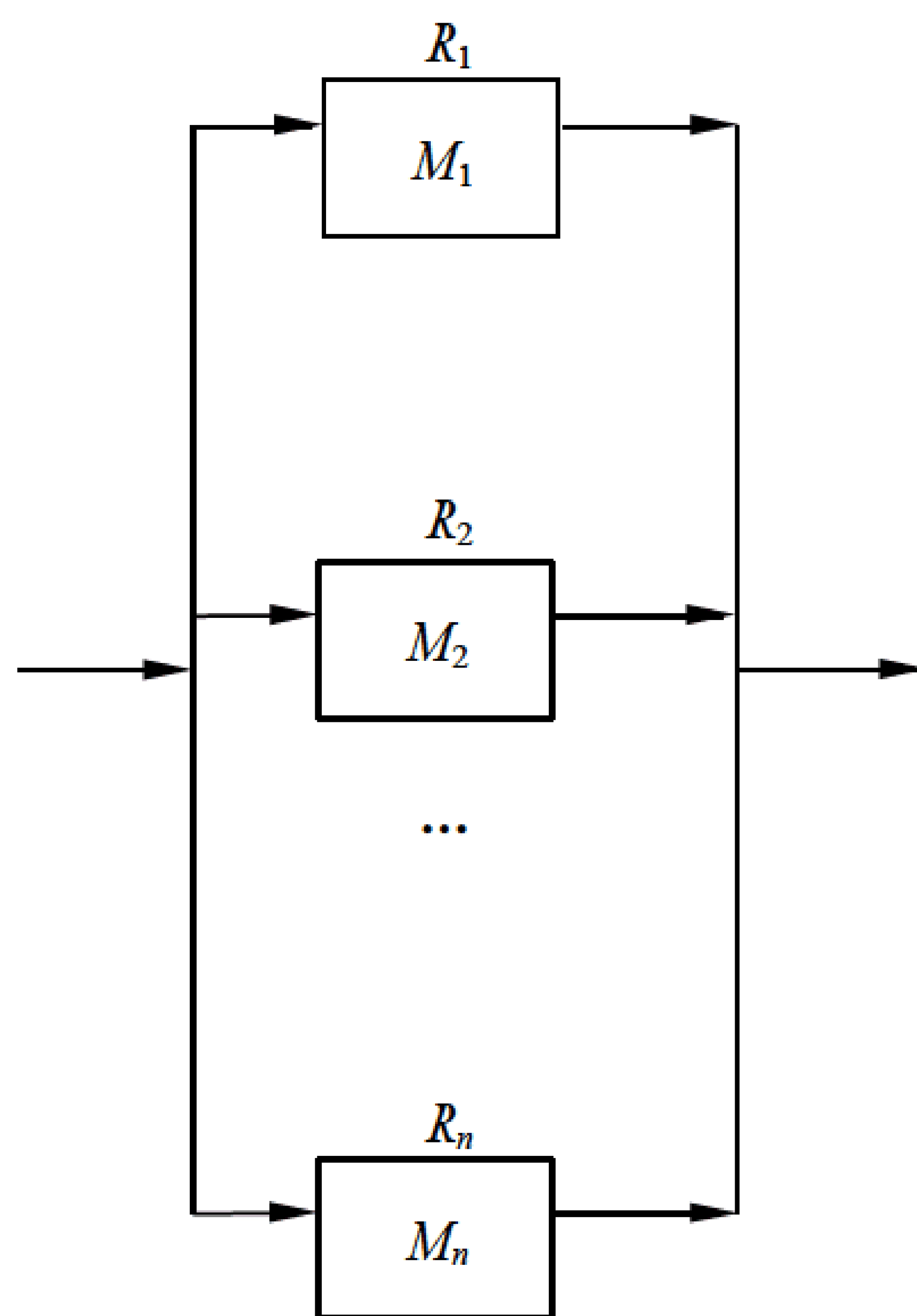


图 7-2　并联系统

并联系统的失效概率为：

$$Q_{sys} = \prod_{i=1}^{n} Q_i$$

并联系统的可靠度为：

$$R_{sys} = 1 - Q_{sys} = 1 - \prod_{i=1}^{n} Q_i = 1 - \prod_{i=1}^{n}(1 - R_i)$$

其中，Q_i=1–R_i 是模块 M_i 的失效概率。

希赛教育专家提示：在本套丛书中的《系统分析师考试全程指导（2009 版）》的第 7.1.3 节也介绍了串联系统和并联系统的可靠度、失效率的计算公式，但那里的公式和本节的公式有点不一样，体现在失效率的计算上。认真去分析二者的差异，可以发现，《系统分析师考试全程指导（2009 版）》的第 7.1.3 节的公式是本节公式的一个简化。本节给出的公式是以时间 t 无穷大为基础进行推导的，而《系统分析师考试全程指导（2009 版）》的第 7.1.3 节的公式是为了计算方便而简化的。

3．串并联系统

如果一个系统由 m 个子系统并联而成，而每个并联的子系统又由 n 个元件串联而成，称这样的系统为串并联系统。

设第 j 个子系统的第 i 个元件的可靠度为 R_{ij} (i=1, 2, …, n；j=1,2,…, m)，则该串并联系统的可靠度为：

$$R_{sp} = 1 - \prod_{j=1}^{m}\left(1 - \prod_{i=1}^{n} R_{ij}\right)$$

如果 R_{ij} 全部相等（假设为 R），则：

$$R_{sp} = 1 - (1 - R^n)^m$$

4．并串联系统

如果一个系统由 n 个子系统串联而成，且其中每个子系统又由 m 个元件并联而成，称这种系统为并串联系统。并串联系统的可靠度为：

$$R_{ps} = \prod_{i=1}^{n}\left\{\left(1 - \prod_{j=1}^{m}(1 - R_{ij})\right)\right\}$$

如果 R_{ij} 全部相等（假设为 R），则：

$$R_{ps} = (1 - (1 - R)^m)^n$$

7.3.3 马尔柯夫模型

马尔柯夫模型的两个核心概念是状态和状态转移。系统的状态表示了在任何瞬间用以描述该系统所必须知道的一切。对于可靠性分析，马尔柯夫模型的每个状态表示了有效和失效模块的不同组合。如果每个模块都是处于有效和失效两种情况之一，则一个 n 模块的系统的完整模型有 2^n 个状态。状态转移是指随着时间的流逝，因模块的失效和修复，系统发生的状态变化。

作为马尔柯夫模型基础的基本假设是：给定状态的转移概率仅取决于当前的状态。系统从一个状态 i 转移到另一个状态 j 的转移率定义为单位时间内从状态 i 转移到状态 j 的概率。对于一个模块来说，从运行状态到失效状态的转移率就是模块的失效率，从失效状态到运行状态的转移率就是模块的修复率。一个失效率为 λ，修复率为 μ 的模块的状态图如图 7-3 所示。

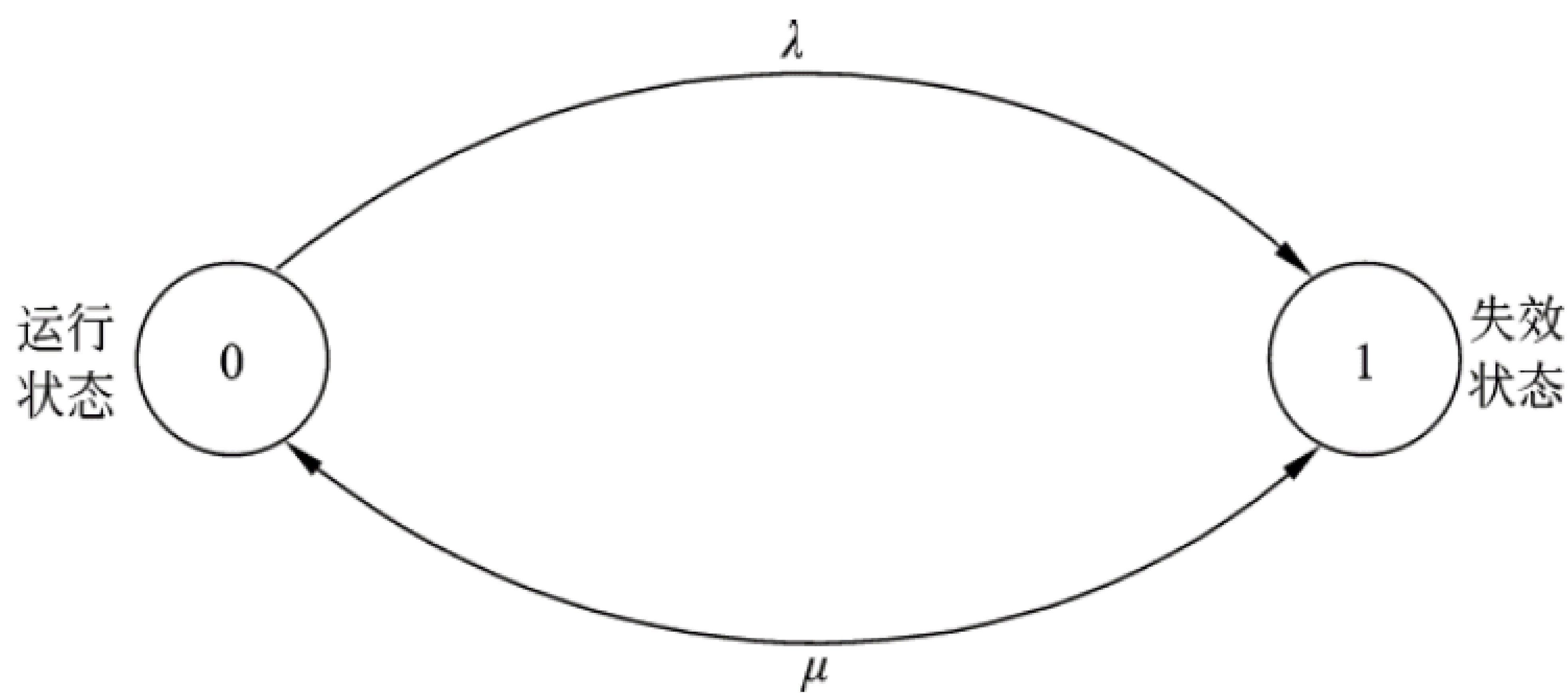


图 7-3 （λ，μ）模块的状态图

从理论上说，任意两个状态之间都存在转移的可能性。但因失效是独立的，在很短的时间内发生多个失效的可能性远小于发生一个失效的可能性。因此，只需要考虑任一时刻只有一个模块失效的转移。同样，也只要考虑任意时刻只有一个模块修复的转移。

系统的状态图也可以表示为层次图。第一层只有一个状态，对应于所有模块都运行的情况；第二层有 n 个状态，对应于一个模块失效的各种情况；第 i+1 层有 C_n^i 个状态，

对应于 n 个模块中有 i 个失效的各种情况；第 n+1 层也只有一个状态，对应于全部模块都失效的情况。

根据系统的状态图，可以计算出系统处于任意状态的概率。设系统在 t 时刻处于状态 0 和 1 的概率分别为 $P_0(t)$和 $P_1(t)$，于是，在 $t+\Delta t$ 时刻系统处于 0 状态的概率为：

$$P_0(t+\Delta t)=P_0(t)-P_0(t)\cdot\lambda\cdot\Delta t+P_1(t)\cdot\mu\cdot\Delta t$$

同样，在 $t+\Delta t$ 时刻系统处于 1 状态的概率为：

$$P_1(t+\Delta t)=P_1(t)+P_0(t)\cdot\lambda\cdot\Delta t-P_1(t)\cdot\mu\cdot\Delta t$$

令 $\Delta t\to 0$ 取极限得微分方程组：

$$\begin{bmatrix}\dot{P}_0(t)\\ \dot{P}_1(t)\end{bmatrix}=\begin{bmatrix}-\lambda & \mu\\ \lambda & -\mu\end{bmatrix}\begin{bmatrix}P_0(t)\\ P_1(t)\end{bmatrix}$$

其中，$\dot{P}_i(t)$ 是 $P_i(t)$ 对 t 的一阶导数($i=0,1$)。只要解此微分方程组就可以得出 $P_0(t)$ 和 $P_1(t)$。

对于有 n 个状态的状态图，设状态 i 到 j 的转移率为 α_{ij} 。考虑其中的任意一个状态 j，其他状态到 j 的转移和 j 到其他状态的转移，系统在 $t+\Delta t$ 时刻，处于状态 j 的概率可以表示为：

$$P_j(t+\Delta t)=P_j(t)-\sum_{\substack{i=1\\ i\neq j}}^{n}(P_j(t)\cdot\alpha_{ji}\cdot\Delta t)+\sum_{\substack{i=1\\ i\neq j}}^{n}(P_i(t)\cdot\alpha_{ij}\cdot\Delta t)$$

由此可得：

$$\dot{P}_j(t)=\sum_{\substack{i=1\\ i\neq j}}^{n}(P_i(t)\cdot\alpha_{ij})-\left(\sum_{\substack{i=1\\ i\neq j}}^{n}\alpha_{ji}\right)\cdot P_j(t)\qquad j=1,2;\cdot n,$$

用矩阵方程把 $\dot{P}_j(t)(j=1,2,\cdots,n)$ 全部表示出来就是：

$$\dot{P}(t)=T\cdot P(t)$$

或：

$$\begin{bmatrix}\dot{P}_1(t)\\ \dot{P}_2(t)\\ \dot{P}_3(t)\\ \vdots\\ \dot{P}_n(t)\end{bmatrix}=\begin{bmatrix}-\beta_1 & \alpha_{21} & \alpha_{31} & \cdots & \alpha_{n1}\\ \alpha_{12} & -\beta_2 & \alpha_{32} & \cdots & \alpha_{n2}\\ \alpha_{13} & \alpha_{23} & -\beta_3 & \cdots & \alpha_{n3}\\ \vdots & \vdots & \vdots & \cdots & \vdots\\ \alpha_{1n} & \alpha_{2n} & \alpha_{3n} & \cdots & -\beta_n\end{bmatrix}\cdot\begin{bmatrix}P_1(t)\\ P_2(t)\\ P_3(t)\\ \vdots\\ P_n(t)\end{bmatrix}$$

其中，**T** 称为状态转移矩阵，其对角线上的元素：

$$\beta_j=\sum_{\substack{i=1\\ i\neq j}}^{n}\alpha_{ji}(j=1,2,\cdots,n)$$

这一矩阵方程称为查普曼-科尔莫戈罗夫（Chapman-Kolmoqorov）方程，由它可解出系统处于任意状态的概率。解方程最常用的是拉普拉斯变换解法。

马尔柯夫模型是计算系统可靠性的强有力工具，用组合模型能计算的可靠性，用马尔柯夫模型也能计算，马尔柯夫模型还能计算许多组合模型不能计算的可靠性。

7.4 提高可靠性的措施

防止故障造成系统失效的两种技术是故障掩蔽技术和系统重组技术。故障掩蔽技术是指防止故障造成差错的各种技术，系统重组技术是防止差错导致系统失效的各种技术。故障掩蔽技术和系统重组技术是达到容错的两种基本途径。而它们又是建立在资源冗余的基础上的。资源冗余有硬件冗余、信息冗余、时间冗余和软件冗余 4 种形式。

本节主要介绍硬件冗余和信息冗余，有关其他冗余技术，请阅读本套丛书中的《系统分析师考试全程指导（2009 版）》的第 7.6.1 节。

7.4.1 硬件冗余

硬件冗余最常用的是三模冗余和 *N* 模冗余。

1. 三模冗余

三模冗余（Triple Modular Redundancy，TMR）是指三个相同的模块接收三个相同的输入，产生的三个结果送至多数表决器。表决器的输出取决于三个输入的多数，若有一个模块故障，则另两个正常模块的输出可将故障模块的输出掩蔽，从而不在表决器输出产生差错。三模冗余的结构如图 7-4 所示。

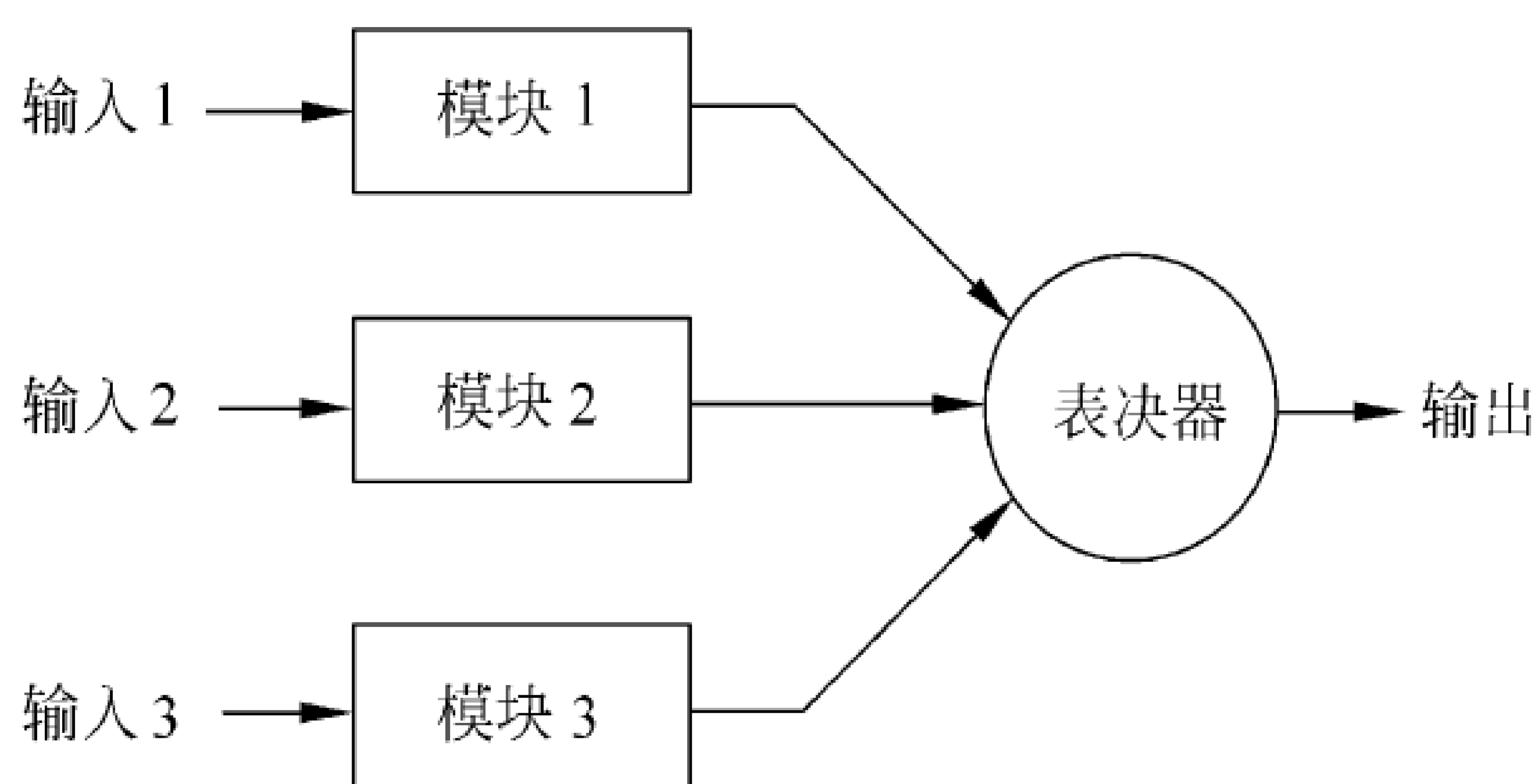


图 7-4 TMR 系统结构图

用组合模型可以计算出系统的可靠度为：

$$R = R_v(R_m^3 + 3R_m^2(1-R_m)) = R_v(3R_m^2 - 2R_m^3)$$

由于 $R_m = e^{-\lambda t}$，因此：

$$R = R_v(3e^{-2\lambda t} - 2e^{-3\lambda t})$$

其中 R_v 和 R_m 分别表示表决器和模块的可靠度（假设各模块的可靠度相同）。

在无修复的屏蔽冗余系统中，当屏蔽冗余因模块中的故障而耗尽时，再发生模块故障将导致输出的错误。假若模块具有修复能力，则系统的可靠性将会大大提高。下面讨论修复对 TMR 结构可靠性的影响。

设模块的失效率为 λ，修复率为 μ，TMR 系统可靠度的马尔柯夫模型如图 7-5 所示。

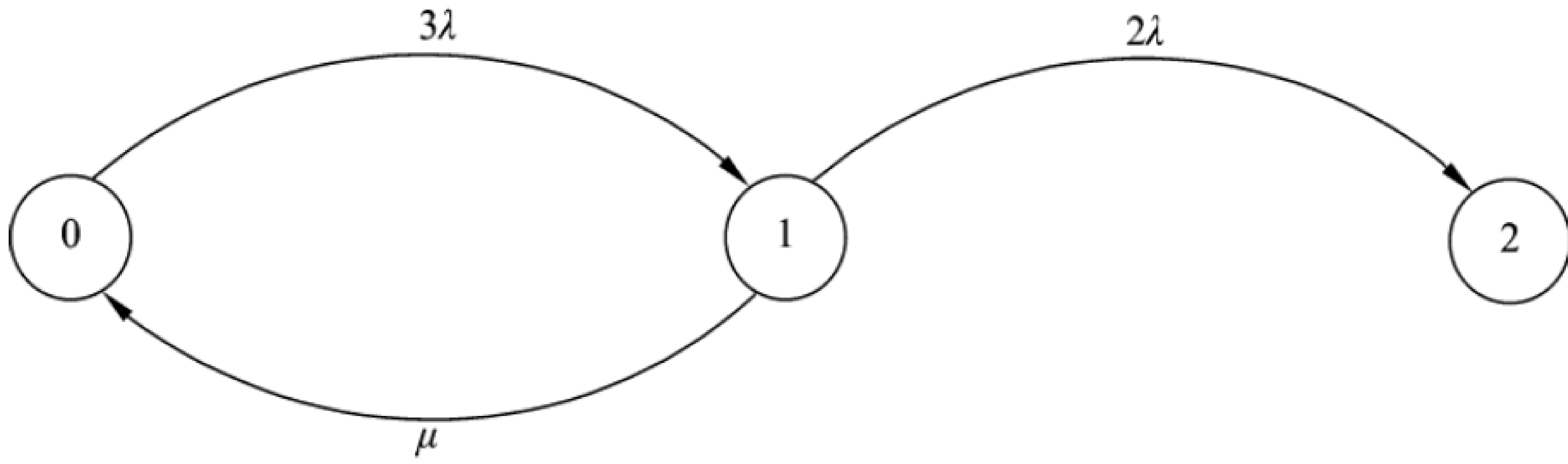


图 7-5 （λ，μ）TMR 的马尔柯夫模型状态图

TMR 系统的拉氏系数矩阵为：

$$\boldsymbol{A}=\begin{bmatrix} s+3\lambda & -\mu & 0 \\ -3\lambda & s+2\lambda+\mu & 0 \\ 0 & -2\lambda & s \end{bmatrix}$$

设系统的初态 $P_0(0)=(100)^T$，则：

$$P_2^*(S)=\frac{\begin{vmatrix} s+3\lambda & -\mu & 1 \\ -3\lambda & s+2\lambda+\mu & 0 \\ 0 & -2\lambda & 0 \end{vmatrix}}{\begin{vmatrix} s+3\lambda & -\mu & 1 \\ -3\lambda & s+2\lambda+\mu & 0 \\ 0 & -2\lambda & s \end{vmatrix}}=\frac{6\lambda^2}{s(s^2+(5\lambda+\mu)s+6\lambda^2)}\approx\frac{6\lambda^2}{s^2(s+5\lambda+\mu)}$$

对上式做部分分式分解，可得：

$$\frac{6\lambda^2}{s^2(s+5\lambda+\mu)}=-\frac{1}{(5\lambda+\mu)^2}\cdot\frac{1}{s}+\frac{1}{(5\lambda+\mu)^2}\cdot\frac{1}{s^2}+\frac{1}{(5\lambda+\mu)^2}\cdot\frac{1}{s+5\lambda+\mu}$$

查拉氏逆变换表可得：

$$P_2(t)\approx-\frac{1}{(5\lambda+\mu)^2}+\frac{t}{(5\lambda+\mu)}+\frac{1}{(5\lambda+\mu)^2}\cdot e^{-(5\lambda+\mu)t}$$

因此，系统的可靠度为：

$$R_{\mathrm{TMR}}(t)=1-P_2(t)\approx1+\frac{1}{(5\lambda+\mu)^2}-\frac{t}{(5\lambda+\mu)}-\frac{1}{(5\lambda+\mu)^2}\cdot e^{-(5\lambda+\mu)t}$$

用同样的方法，可以计算出带修复的 TMR 的可用度 $A_{\mathrm{TMR}}(t)$。

由 TMR 的完全状态图，可以计算出系统的拉氏系数矩阵为：

$$\boldsymbol{A}=\begin{bmatrix} s+3\lambda & -\mu & 0 & 0 \\ -3\lambda & s+2\lambda+\mu & -2\mu & 0 \\ 0 & -2\lambda & s+2\mu+\lambda & -3\mu \\ 0 & 0 & -\lambda & s+3\mu \end{bmatrix}$$

用马尔柯夫模型可解出：

$$P_0(t)=\frac{\mu^3+3\lambda\mu^2 e^{-(\lambda+\mu)t}+3\lambda^2\mu e^{-2(\lambda+\mu)t}+\lambda^3 e^{-3(\lambda+\mu)t}}{(\lambda+\mu)^3}$$

$$P_1(t)=\frac{3\lambda\mu^2+3\lambda\mu(2\lambda-\mu)e^{-(\lambda+\mu)t}+3\lambda^2(\lambda-2\mu)e^{-2(\lambda+\mu)t}-3\lambda^3 e^{-3(\lambda+\mu)t}}{(\lambda+\mu)^3}$$

$$P_2(t)=\frac{3\lambda^2\mu+3\lambda^2(\lambda-2\mu)e^{-(\lambda+\mu)t}-3\lambda^2(2\lambda-\mu)e^{-2(\lambda+\mu)t}+3\lambda^3 e^{-3(\lambda+\mu)t}}{(\lambda+\mu)^3}$$

$$P_3(t)=\frac{\lambda^3-3\lambda^3 e^{-(\lambda+\mu)t}+3\lambda^3 e^{-2(\lambda+\mu)t}-\lambda^3 e^{-3(\lambda+\mu)t}}{(\lambda+\mu)^3}$$

则 TMR 结构系统的可用度为：

$$A_{\mathrm{TMR}}(t)=P_0(t)+P_1(t)$$

对 TMR 函数积分，可得出平均无故障时间：

$$\mathrm{MTTF}=\frac{5\lambda+\mu+\sqrt{\lambda^2+10\lambda\mu+\mu^2}}{(5\lambda+\mu)\sqrt{\lambda^2+10\lambda\mu+\mu^2}-\lambda^2-10\lambda\mu-\mu^2}-\frac{5\lambda+\mu-\sqrt{\lambda^2+10\lambda\mu+\mu^2}}{(5\lambda+\mu)\sqrt{\lambda^2+10\lambda\mu+\mu^2}+\lambda^2+10\lambda\mu+\mu^2}$$

简化得：

$$\mathrm{MTTF}=\frac{5}{6\lambda}+\frac{\mu}{6\lambda^2}$$

上面的分析没有考虑表决的可靠性，为了能够容忍表决器的故障，可以对表决器也采用三倍冗余。一般说来，对于一个由若干模块构成的系统，可以对每个模块分别实施 TMR 技术。

2．*N* 模冗余

TMR 的推广是 *N*（*N*≥3）模冗余（N-Modular Redundancy，NMR）。与三模冗余原理相同，但采用 *N* 个相同的模块且 *N* 为奇数，以方便进行多数表决。NMR 的结构如图 7-6 所示。

7.4.2 信息冗余

信息冗余是指通过在数据中附加冗余的信息以达到故障检测、故障掩蔽或容错的目的。应用最广泛的是奇偶校验码、海明校验码。

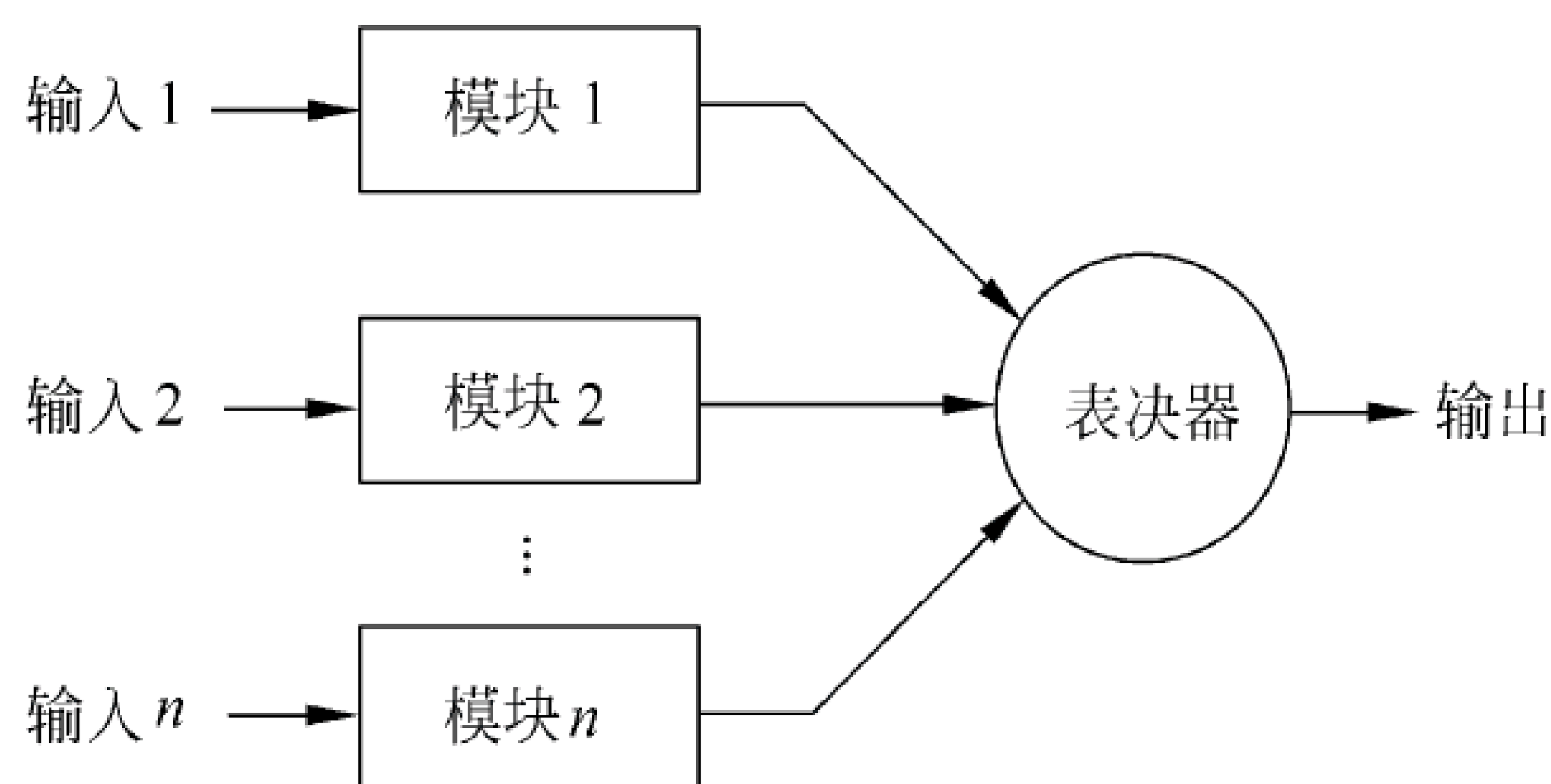


图 7-6　NMR 系统结构图

基本的海明纠错码能纠正一位错。它的原理是基于重叠奇偶校验的概念：将原始数据位分成若干个重叠的组，每组设一位奇偶校验位。由于组间有重叠，因此每位原始数据从属于多于一个组。而且每位原始数据的从属关系是不一样的，纠错时，根据哪些组的奇偶校验位出错，就可以唯一地确定是哪一位数据出错。将该位取反就完成了纠错。

海明码是一种特殊的（n，k）线性纠错码，线性纠错码可借助它们的奇偶校验矩阵来描述。（n，k）线性码的奇偶校验矩阵是一个$(n–k)\times n$ 的矩阵，其元素是 0 和 1。矩阵的每一列与码字中的一个位相对应，而每一行与校验位相对应。

本章参考文献

[1] 胡谋. 计算机容错技术. 北京：中国铁道出版社，1996

[2] 袁由光，杨孝宗. 可靠系统的理论与实践（上下册）. 北京：科学出版社，1993

[3] 袁由光. 实时系统中的可靠性技术. 北京：清华大学出版社，1995

[4] 黄锡滋. 软件的可靠性与安全性. 北京：清华大学出版社，2001

[5] 刘斌，高小鹏. 嵌入式软件可靠性仿真测试系统研究. 北京航空航天大学学报，2000（8）：490-493

第 8 章　系统的安全性和保密性设计

当今的信息系统基本上都是基于网络的系统，特别是随着 Internet 的发展和普及，以后的系统将绝大部分基于互联网运行，使企业能够与他们的合作伙伴共享信息，打造完整的供应链，同时，也便于企业实施电子商务工程。在基于网络运行的系统中，一个很重要的设计问题，就是要考虑系统和数据的安全性和保密性。

8.1　信息安全概述

只要网络存在，信息安全就是一个永恒的话题。没有一个系统可以说是完全安全的，只是一个攻击的代价问题。“魔高一尺，道高一丈；魔再高一尺，道再高一丈”，如此循环往复，促进了信息安全技术的发展。

8.1.1　信息安全概念的发展

信息安全是一个不断发展的概念。自 1949 年 Shannon 发表《保密通信的信息理论》开始，对信息安全的研究全面展开。可以认为，到目前为止，对信息安全的研究经历了 4 个阶段。

1．通信保密阶段

这一阶段一直持续到 20 世纪 70 年代。在这一阶段中，主要的安全威胁是搭线监听。因此，人们关心的是信息传递的问题，即如何保证信息传递过程中机密数据不会为第三方所知。所采用的方法包括加密、传输保密、发射保密，以及计算机的物理安全，研究重点是通过密码（主要是序列密码）解决通信保密问题。可以说在这一阶段中，人们所关注的是信息机密性和信息收发真实性的问题。

2．计算机安全阶段

美国国家标准局 1977 年公布了国家数据加密标准（Data Encryption Standard，DES），美国国防部 1983 年公布了可信计算机系统评价准则（Trusted Computer System Evaluation Criteria，TCSEC），从而标志着计算机安全的研究进入新的阶段。

20 世纪的七八十年代，对信息安全的研究进入了一个崭新的阶段，这一阶段的许多成果成为目前信息安全研究的基础。在这一阶段，人们对信息安全问题有了更全面的认识，将信息安全的研究归纳为三个方面：机密性（confidential）、完整性（integrity）和可用性（availability），即著名的 CIA 模型，时至今日，这三个方面仍是信息安全的三个主要问题。

Anderson 在 1972 年提出了引用监视器（Reference Monitor，RM）和引用确认机制（Reference Validation Mechanism，RVM）的概念，在此基础上，人们又提出了 TCB（Trusted Computing Base，可信计算基础）的概念，成为目前可信计算的基础理论。在 Anderson 报告建立的框架下，Bell 和 Lapadula 根据美国军方的实际需求，模拟军方手工保护机密文件的政策，给出了一个著名的形式化模型（Bell-Lapadula 模型），该模型重点是解决信息的保密性，模型利用集合论和关系代数，定义系统及系统状态，在数学上证明了符合模型的系统是安全的。有关 Bell-Lapadula 模型的详细知识，请阅读第 8.2.3 节。

3. 信息安全阶段

20 世纪 90 年代以来，通信和计算机技术相互依存，数字化技术促使计算机网络发展成为全天候、通全球、个人化、智能化的信息高速公路，Internet 成了寻常百姓可及的家用技术平台，信息安全的概念随之产生，安全的需求不断地向社会的各个领域扩展。人们需要保护信息在存储、处理或传输过程中不被非法访问或更改，以及确保对合法用户的服务和对非授权用户服务的限制，包括必要的检测、记录和抵御攻击的措施。此阶段对安全性有了新的需求。

这一时期，在密码学方面，公开密钥密码得到了巨大的发展，著名的 RSA（Ron Rivest，AdiShamir 和 Leonard Adleman）公开密钥密码算法获得了日益广泛的应用，用于完整性校验的 Hash 函数的研究应用也越来越多。为了奠定 21 世纪的分组密码算法基础，美国国家技术标准研究所推行了高级加密标准项目。而更强、更快的公开密钥密码算法研究和应用，则把希望寄托在椭圆曲线公开密钥密码算法上。目前，安全威胁已发展到黑客的网络入侵、病毒破坏、计算机犯罪事件等。

4. 信息保障阶段

时至今日，对于信息系统的攻击日趋频繁，安全的概念已经不再局限于信息的保护，人们需要的是对整个信息和信息系统的保护和防御，以确保它们的安全性，这包括了对信息的保护、检测、反应和恢复能力，这就是信息安全保障的概念。为了保障信息安全，除了要进行信息的安全保护，还应该重视提高系统的入侵检测能力、系统的事件反应能力和系统遭到入侵引起破坏时的快速恢复能力。区别于传统的加密、身份认证、访问控制、防火墙、安全路由等技术，信息保障强调信息系统整个生命周期的防御和恢复。

8.1.2　信息安全研究的目标

随着对信息安全研究的不断深入，信息安全的内容也在不断发展。除了传统的 CIA 以外，真实性、不可否认性、系统可存活性等问题也已经成为信息安全的研究目标。

真实性指的是数据或服务访问者身份的确定性或者匿名访问的合法性。随着网络范围的不断扩张，人们希望可以坐在任何一台计算机前访问被授权的服务，身份验证就成了网络服务所必须考虑的问题。除此之外，广泛的信任机制与分布式授权访问也成为信息安全的研究问题之一。

不可否认性也叫做不可抵赖性，它随着电子商务的发展也得到更多的重视。某些消息要求消息的制造者不能否认自己的身份，他们必须为自己的消息承担相应的责任。采用公钥算法体系的电子签名是确保不可否认性的重要手段。

美国国防部组织了一系列的信息安全项目对系统可存活性展开了研究。随着对软件开发的认识越来越理性化，人们发现开发一个完全没有缺陷的系统几乎是不可能的，而且代价颇为昂贵；随着软构件的发展，由于可以大幅度地降低开发造价，商用货架产品（Commercial Off The Shelf，COTS）也在关键系统与专用系统中得到大量的应用，这些商用构件的固有缺陷更难以弥补。如何提高这类有缺陷系统的安全性就成为一个重要的问题。入侵容忍系统就是一种旨在提高系统可存活性的系统，这种系统通过表决、复制、重配等手段使得系统即使已经遭受了攻击或出现了故障也能够提供正常的或降级的服务，保证系统仍然可以继续工作。

表 8-1 中给出了各安全属性的概念及保护方法举例。

表 8-1　安全属性对比表

研 究 目 标	定　　义	保护手段举例
机密性（Confidentiality）	防止未经授权的数据访问能力的度量	访问控制 数据加密
完整性（Integrity）	防止数据被篡改能力的度量	报文摘要
可用性（Availability）	衡量用户在规定时间内以规定的方式访问服务的能力	可靠性设计 冗余备份
真实性（Authenticity）	对数据访问者的真实身份鉴别能力的度量	身份识别
不可否认性（Non- reputation）	消息制造者无法否认消息归属能力的度量	数字签名
可存活性（Survivability）	衡量系统遭受入侵后提供持续服务的能力	入侵容忍

8.1.3　信息安全的常用技术

除了对数据进行加密传输外，常用的安全技术有访问控制、防火墙、入侵检测、入侵容忍等。

1．访问控制

访问控制是一种经典的安全手段，至今仍在使用。人们通常使用一张访问控制表定义用户对资源的访问权限，来保护数据不被泄漏或被篡改。有关访问控制方法和控制模型的详细知识，请阅读第 8.2 节。

2．防火墙

防火墙是一种非常有效的阻隔入侵的方法，防火墙也沿袭了访问控制的思想，对网络外部的访问进行筛选和过滤，仅仅放行规则允许的数据包。有关防火墙技术的详细知

识，请阅读第 8.5.3 节。

3．入侵检测

防火墙固然能够防御很多恶意攻击，但防火墙有两大缺点。第一，防火墙无法防御系统正常服务所具有的漏洞，如果系统提供的网络服务具有漏洞（WWW 服务的漏洞极为常见），防火墙不会拦截到这些符合规则的入侵指令；第二，防火墙仅能应对外部攻击，而对内部发起的入侵无能为力。为解决防火墙的这些问题，入侵检测技术应运而生。

入侵检测技术初衷是希望在入侵发生之前，检测到入侵的征兆，并将入侵消除在初始阶段。入侵检测技术经过十余年的发展已日趋成熟，一些商用的入侵检测系统也投入使用。总的来说，入侵检测技术可以分为 4 大类。

（1）模式匹配。模式匹配的方法为最初的入侵检测系统广为使用，其基本思想类似于病毒检测，通过对网络中通行的数据包进行数据特征的检验，寻找与模式库（入侵库）中特征性匹配的数据包，并将这种数据包看成入侵数据包。这种方法实现起来非常简单，但漏报率和误报率都比较高，而且算法效率不高，在网络带宽已经以 Gbps 计算的今天显得有些力不从心。除此之外，这种方法对模式库有很高的要求，如果模式库陈旧就不能及时发现入侵，让入侵从鼻子底下溜过去。同时，如果狡猾的入侵者将一条入侵指令拆分为几个部分，每一部分都不包含完整的特征，模式匹配方法也不能做出有效的反应。目前，几乎没有哪种系统采用单纯模式匹配的方式进行入侵检测了。

（2）概率分析。正常的用户在工作的时候总是遵循一定的顺序或规则，如果将某一时间周期内用户输入的指令看作一个整体，那么就会发现这个整体有一定的规律可循，或者说这个整体的模式依概率收敛；而入侵者则和正常用户恰恰相反，尤其是在发现系统脆弱性的时候，入侵者总是做出各种怪异的行为。这就给安全工作者提供了一种思路，通过对用户行为特征的检测和分析来确定入侵，这种方法叫做误用检测。除用户行为外，系统行为或者说是系统的内部状态的变化也是有规律可循的，对于系统状态的变化规律的检测和分析叫做异常检测。无论是误用检测还是异常检测都是以概率为基础，建立正确的概率模型，并对非正常的行为做出分析和判断。这种方法首先是效率不高，其次是很难正确的建立系统模型。因此，基于概率分析的入侵检测技术没有得到太多的应用，仍然处在理论阶段。

（3）协议分析。协议分析是模式匹配的发展，但大大提高了检测的效率。模式匹配固然简单，但简单匹配的方法其实作了很多无用功。对于文件传输协议（File Transfer Protocol，FTP）服务的攻击方法肯定不会用于超文本传输协议（Hypertext Transfer Protocol，HTTP）服务，更进一步，如果发送给 FTP 服务的数据包中出现了 HTTP 攻击的特征，那肯定是一次误报。基于这个思路，人们对模式匹配的方法进行了改进，对入侵数据包进行了协议分析。首先分析接收到的数据包采用的协议，然后再有计划地进行特征查找，不但提高了效率，同时也减少了误报率。由于协议分析的方法具有检测率和检测效率都比较高的优点，很多商用入侵检测系统采用了协议分析的方法。

希赛教育专家提示：目前，学术界还在不断探索新的入侵检测方法。例如，使用神经元网络进行入侵检测就是其中之一，神经元网络在模糊分类和模式识别方面有着不错的效果。通过对其反复训练，神经元网络也能具有入侵检测的能力。对于很多入侵来说，其步骤可能是复杂的，采用的不是一个简单的数据包而是一个攻击序列。因此，有人提出使用 Petri 网或着色 Petri 网进行入侵检测，在这方面也有相关的论文出现。这些方法都是对入侵检测技术有益的探索，但目前还都不够成熟。

采用入侵检测技术构造的系统也分为两大类。一类是主机性入侵检测系统（Host-based Intrusion Detection System，HIDS），一类是网络型入侵检测系统（Network Intrusion Detection System，NIDS）。HIDS 仅仅关心某台主机是否遭受了入侵，而 NIDS 则会动态监视网络中的数据，力求不放过每一个入侵企图。

4．入侵容忍

即使有访问控制、防火墙和入侵检测系统的多层保护，但入侵事件不但没有下降反而不断地上升。也就是说，使用访问控制、防火墙和入侵检测技术保护的系统并不是那么安全。

访问控制是一种静态的安全技术，防火墙的特点是防外不防内，而入侵检测系统对于未知的攻击则无能为力。因此，需要一种动态的、可以为系统提供全面保护的、可以有效防范未知攻击的安全措施。入侵容忍技术是一种发展中的技术，它的目标并不是在入侵发生之前就将攻击拒之门外，而是保护系统即使遭受了入侵，仍然不会停止服务或服务异常，这非常类似于容错技术，这是入侵容忍技术的出发点。

为达到容忍入侵的目标，入侵容忍技术中通常包括冗余策略和恢复策略，这些策略与容错技术采用的冗余恢复的方法是略有不同的。更确切地说，入侵容忍技术采用了多样性冗余和快速恢复的策略。所谓多样性冗余指的是冗余部件与工作部件不完全一样，如果导致工作部件失效的入侵也能够导致冗余部件失效，那么无论恢复的速度又多快，系统最终还是不可避免的失效。多样性冗余可以分为设计多样性和实现多样性，构造异种平台或 N 版本程序都是多样性冗余的典型例子。快速恢复策略指的是在很短的时间内（时间短到保证服务的持续性）恢复系统，哪怕是不完全恢复，仅仅提供一个降级的服务也可以。

入侵容忍技术的研究范围最少要包含三个子问题：入侵发现、入侵处理与灾难恢复。

入侵发现不同于入侵检测，二者有着明显的不同。入侵发现和入侵检测最大的区别在于发生时机和误检率。一般来说，入侵检测系统的目的是阻隔入侵，因此它总是试图在入侵发生之前就检测到入侵行为，模式匹配或推断逻辑非常重要，误检率较高；而入侵发现的目的是判断系统是否已经遭受了入侵从而决定是否要启动入侵处理与恢复机制，因此其发生时机在入侵发生之后，检测手段较为简单，误检率很低。除此之外，常用的入侵检测系统多是基于模式匹配的方法，着眼于检测出某种特定的攻击模式或入侵行为。虽然某些入侵检测系统通过误用检测甚至人工神经元网络的手法试图发现入侵事

件，但其着眼点还是某一类特定的入侵行为。而我们所说的入侵发现着眼于广泛的入侵行为，主要通过检测系统的异常来判断入侵是否已经发生。

入侵处理是当发现入侵后，入侵容忍系统需要对入侵行为进行一系列的处理，以防止入侵灾难的进一步扩大。入侵处理不同于灾难恢复，灾难恢复的重点是恢复已经遭受入侵的服务、资源、组件或进程，而入侵处理的目的是防止入侵行为带来的危害进一步扩大，这一步是不可或缺的。如果不进行入侵处理，待发现入侵之后单纯的进行恢复工作可能会让系统陷入不断的被入侵和持续恢复的循环中，最终当恢复的速度小于入侵速度或者由于系统资源完全耗尽等原因无法进行恢复时，系统仍将崩溃。

灾难恢复指的是将已经被破坏的系统资源和系统部件恢复到正常或降级的工作状态中，保证服务的持续性。灾难恢复可以分为前向恢复、后向恢复和补偿恢复的办法。无论采用哪一种恢复办法，都要注意恢复的时间长度，这对于入侵容忍技术非常重要。

8.2　访问控制技术

ISO 所定义的五大安全服务功能是：认证（authentication）、访问控制（Access Control）、数据保密性（Data Confidentiality）、数据完整性（Data integrity）和防止否认（non-reputation）服务，其中访问控制服务在系统安全体系结构中起着不可替代的作用。

访问控制是通过某种途径显式地准许或限制访问能力及范围的一种方法。它是针对越权使用资源的防御措施，通过限制对关键资源的访问，防止非法用户的侵入或因为合法用户的不慎操作而造成的破坏，从而保证系统资源受控地、合法地使用。用户只能根据自己的权限大小来访问系统资源，不得越权访问。访问控制技术通常和身份认证密切联系，但并不能取代身份认证，它是建立在身份认证的基础之上的，通俗地说，身份认证解决的是“你是谁，你是否真的是你所声称的身份”，而访问控制技术解决的是“你能做什么，你有什么样的权限”这个问题，它们在安全系统中所处的位置如图 8-1 所示。

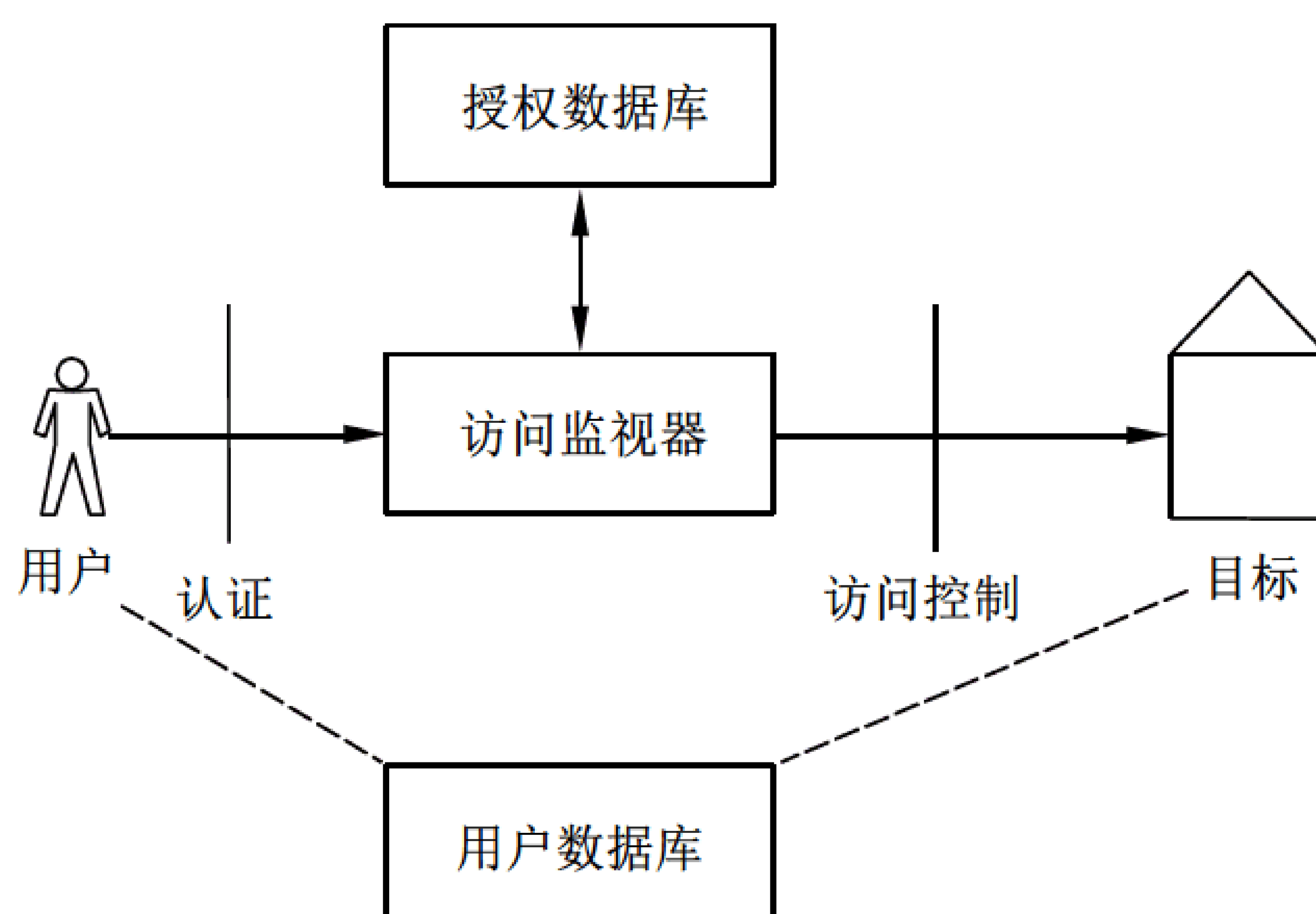


图 8-1　一个安全系统的逻辑模型

访问控制系统一般包括主体（subject）、客体（object）、安全访问政策几个实体。访问控制的目的是：限制主体对访问客体的访问权限，从而使计算机系统在合法范围内使用；决定用户能做什么，也决定代表一定用户利益的程序能做什么。

8.2.1 访问控制的实现方法

访问控制的常见实现方法有访问控制矩阵、访问能力表、访问控制表和授权关系表等几种。

1．访问控制矩阵

从数学角度看，访问控制可以表示为一个矩阵的形式，其中行表示客体（各种资源），列表示主体（通常为用户），行和列的交叉点表示某个主体对某个客体的访问权限（如读、写、执行、修改、删除等）。

2．访问能力表

访问控制矩阵虽然直观，但并非每个主体和客体之间都存在着权限关系，相反，实际的系统中虽然可能有很多的主体和客体，但主体和客体之间的权限关系可能并不多，这样的话就存在着很多的空白项。为了减轻系统开销与浪费，可以从主体（行）出发，用访问能力表描述矩阵某一行的信息；也可以从客体（列）出发，用访问控制表（Access Control List，ACL）描述矩阵某一列的信息。

能力（capability）是受一定机制保护的客体标志，标记了客体以及主体（访问者）对客体的访问权限。只有当一个主体对某个客体拥有访问的能力时，它才能访问这个客体。

在访问能力表中，由于它着眼于某一主体的访问权限，以主体的出发点描述控制信息，所以很容易获得一个主体所被授权可以访问的客体及其权限，但如果要求获得对某一特定客体有特定权限的所有主体就比较困难。在一个安全系统中，正是客体本身需要得到可靠的保护，访问控制服务也应该能够控制可访问某一客体的主体集合，能够授予或取消主体的访问权限，于是出现了以客体为出发点的实现方式（访问控制表），现代的操作系统大都采用基于访问控制表的方法。

3．访问控制表

ACL 是目前采用最多的一种实现方式。它可以对某一特定资源指定任意一个用户的访问权限，还可以将有相同权限的用户分组，并授予组的访问权。

ACL 的优点在于它的表述直观、易于理解，而且比较容易查出对某一特定资源拥有访问权限的所有用户，有效地实施授权管理。在一些实际应用中还对 ACL 做了扩展，从而进一步控制用户的合法访问时间，是否需要审计等。

尽管 ACL 灵活方便，但将它应用到网络规模较大、需求复杂的企业内部网络时，就暴露了一些问题：

（1）ACL 需对每个资源指定可以访问的用户或组以及相应的权限。当网络中资源很

多时，需要在 ACL 设定大量的表项。而且，当用户的职位、职责发生变化时，为反映这些变化，管理员需要修改用户对所有资源的访问权限。另外，在许多组织中，服务器一般是彼此独立的，各自设置自己的 ACL，为了实现整个组织范围内的一致的控制政策，需要各管理部门的密切合作。所有这些，使得访问控制的授权管理变得费力而繁琐，且容易出错。

（2）单纯使用 ACL，不易实现最小权限原则及复杂的安全政策。

4. 授权关系表

授权关系表（Authorization Relations）弥补了基于 ACL 和基于访问能力表的方法的不足，它的每一行（或称一个元组）表示了主体和客体的一个权限关系。如果该表按客体进行排序就拥有访问能力表的优势，如果按主体进行排序就拥有了访问控制表的好处。这种实现方式也特别适合采用关系数据库。

8.2.2　访问控制策略

自主访问控制（Discretionary Access Control，DAC）、强制访问控制（Mandatory Access Control，MAC）和基于角色的访问控制（Role-Based Access Control，RBAC）是系统中通常采用的三种不同的访问控制策略，其中 DAC 和 MAC 属于传统的访问控制策略，而 RBAC 则是 20 世纪 90 年代后涌现出来的新的访问控制策略，由于优势很大，所以发展很快。

1. 自主访问控制

DAC 是目前计算机系统中实现最多的访问控制机制，它是在确认主体身份以及（或）它们所属组的基础上对访问进行限制的一种方法，称其为自主型是因为在 DAC 系统中，一个拥有一定访问权限的主体可以直接或间接地将权限传给其他主体。其基本思想是：允许某个主体显式地指定其他主体对该主体所拥有的信息资源是否可以访问以及可执行的访问类型。Windows、UNIX 系统都是采用了自主型的访问控制技术。

自主访问控制根据访问者的身份和授权来决定访问模式，主体访问者对访问的控制有一定的权利。但正是这种权利使得信息在移动过程中其访问权限关系会被改变。如用户 A 可以将其对客体目标 O 的访问权限传递给用户 B，从而使不具备对 O 访问权限的 B 也可以访问 O，这样就很容易产生安全漏洞，所以自主访问控制的安全级别很低。

随着网络的迅速发展扩大，尤其是 Internet 的兴起，对访问控制服务的质量也提出了更高的要求，传统的 DAC 已很难满足要求。首先，如上所述，DAC 将赋予或取消访问权限的一部分权利留给用户个人，管理员难以确定哪些用户对哪些资源有访问权限，不利于实现统一的全局访问控制。其次，在许多组织中，用户对他所能访问的资源并不具有所有权，组织本身才是系统中资源的真正所有者。而且，各组织一般希望访问控制与授权机制的实现结果能与组织内部的规章制度一致，并且由管理部门统一实施访问控制，不允许用户自主地处理。显然 DAC 已不能适应这些需求。

2. 强制型的访问控制

强制访问控制是“强加”给访问主体的，即系统强制主体服从访问控制政策。MAC 主要用于多层次安全级别的军事应用当中，它预先将主体和客体分级，即定义用户的可信任级别及信息的敏感程度（安全级别，比如可以分成绝密级、机密级、秘密级、无密级等），然后根据主体和客体的级别标记来决定访问模式，用户的访问必须遵守安全政策划分的安全级别的设定以及有关访问权限的设定。当用户提出访问请求时，系统对主客体两者进行比较以确定访问是否合法。

在典型应用中，MAC 的访问控制关系可以分为两种：用上读/下写来保证数据完整性以及利用下读/上写来保证数据的保密性。他们都是通过梯度安全标签实现信息的单向流通。

MAC 最主要的优势在于它能阻止特洛伊木马。一个特洛伊木马是在一个执行某些合法功能的程序中隐藏的代码，它利用运行此程序的主体权限违反安全策略，通过伪装成有用的程序在进程中泄漏信息。一个特洛伊木马能够以直接与非直接泄漏两种方式泄漏信息。前者，特洛伊木马使信息的安全标示不正确并泄漏给非授权用户；后者特洛伊木马通过在返回给一个主体的合法信息中编制的方式非直接泄露信息，例如，可能表面上某些提问需要回答，而实际上用户回答的内容被传送给特洛伊木马。

阻止特洛伊木马的策略是基于非循环信息流，所以在一个级别上读信息的主体一定不能在另一个违反非循环规则的安全级别上写。同样，在一个安全级别上写信息的主体一定不能在另一个违反非循环规则的安全级别上读。由于 MAC 策略是通过梯度安全标签实现信息的单向流通，从而它可以很好地阻止特洛伊木马的泄密。

MAC 的主要缺陷在于实现工作量太大，管理不便，不够灵活。而且，MAC 由于过于偏重保密性，对其他方面如系统连续工作能力、授权的可管理性等考虑不足。

3. 基于角色的访问控制

RBAC 技术可以有效地克服传统访问控制技术中存在的不足之处，可以减少授权管理的复杂性，降低管理开销。而且，RBAC 还能为管理者提供一个比较好的安全实现政策的环境，成为了实施面向企业的安全策略的一种有效的访问控制方式。

角色是一个或一群用户在组织内可执行的操作组合。用户在一定的部门中具有一定的角色，其所执行的操作与其所扮演的角色的职能相匹配，这正是基于角色的访问控制的基本特征。即依据 RBAC 策略，系统定义了各种角色，每种角色可以完成一定的职能，不同的用户根据其职能被赋予相应的角色，一旦某个用户成为某角色的成员，则此用户可以完成该角色所具有的职能。

RBAC 根据用户在组织内所处的角色进行授权与访问控制。也就是说，传统的访问控制直接将访问主体（发出访问操作、存取要求的主动方）和客体（被调用的程序或欲存取的数据访问）相联系，而 RBAC 在中间加入了角色，通过角色沟通主体与客体。在 RBAC 中，用户标识对于身份认证以及审计记录是十分有用的，但真正决定访问权限的

是用户对应的角色标识。由于用户与客体无直接联系，他只有通过角色才享有该角色所对应的权限，从而访问相应的客体，因此用户不能自主地将访问权限授给别的用户，这是 RBAC 与 DAC 的根本区别所在。RBAC 与 MAC 的区别在于：MAC 是基于多级安全需求的，而 RBAC 则不是，因为军用系统中主要关心的是防止信息从高安全级流向低安全级，即限制“谁可以读/写什么信息”，而基于角色控制的系统中主要关心的是保护信息的完整性，即“谁可以对什么信息执行何种动作”。角色控制比较灵活，根据配制可以使某些角色接近 DAC，而某些角色更接近于 MAC。

角色由系统管理员定义，角色成员的增减也只能由系统管理员来执行，即只有系统管理员有权定义和分配角色，而且授权规定是强加给用户的，用户只能被动接受，不能自主地决定，用户也不能自主地将访问权限传给他人，这是一种非自主型访问控制。

最后，要指出的是，角色和组的区别。组通常仅仅是作为用户的集合，而角色一方面是用户的集合，另一方面又是权限的集合，作为中间媒介将用户和权限连接起来。当然，角色可以在组的基础上实现，这样就对保持原有系统非常有利。此时角色就成为一个策略部件，与组织的授权、责任关系相联系，而组成为实现角色的工具，两者间是策略与实现机理的关系。

总之，RBAC 的优势在于：便于授权管理；便于角色划分；便于赋予最小权限原则；便于职责分离；便于客体分类。

希赛教育专家提示：以上每种策略不是绝对互斥的，可以把几种策略综合起来应用从而获得更好、更安全的系统保护。当使用多重策略的时候，只有各种策略的交集策略被允许，访问才被许可。当然，在某些场合下，也可能存在着一些冲突。例如，被某一策略许可的访问被另一策略所禁止，这样就产生冲突，这种情况需要在管理层通过协商来协调。

8.2.3　Bell–Lapadula 模型

Bell-Lapadula 模型是一个经典的形式化模型，该模型利用集合论和关系代数定义系统及系统状态，在数学上证明了符合模型的系统是安全的。

在模型中包含三个部分，分别是主体集合、客体集合和访问控制矩阵。模型中定义了一系列的安全级别，每一个主体和客体都被指定一个确定的安全级别。例如，无级别<秘密<机密<绝密。

对于每一个主体来说，定义一个对客体的访问权限的集合，这些权限包括以下一些：

（1）Read Only：主体仅对客体拥有读权限，不可写。

（2）Append：主体仅对客体添加有些权限，但不可读。

（3）Execute：主体可以执行客体，但不可写也不可读。

（4）Read-Write：主体对客体既可写也可读。

于是，可以通过安全级别来确定唯一的权限。Bell-Lapadula 模型中定义了以下几条

访问准则：

（1）一个高级别的主体可以向下读一个级别低于或等于他的客体，所有最高级别的主体可以读取整个系统的全部客体。

（2）一个低级别的客体不能够读取高级别的主体。

（3）当主客体位于相同层次时，主体对客体的权限为 Read-Write。

（4）一个低级别的主体对级别高于或等于它的客体拥有 Append 权限，则可以根据第 1 条规则来推得这一点。

（5）一个高级别的主体不能够向下写低于或等于它的客体。

归纳起来，该模型的规则为 Read-Down 和 Write-Up。可以证明，完全遵循该模型制定访问控制策略的系统是安全的。

希赛教育专家提示：严格按照 Bell-Lapadula 模型建立起来的系统存在着信息安全级别不断上升的趋势，如果这种上升不断地继续下去，最终将导致全部信息都处在最高的安全级。这种现象意味着用户必须授予越来越高的安全许可才能够得到信息，因此，分级保护也就失去了意义。解决这一问题的方法是引入可信用户（trusted user），并授予可信用户“向下写”的权力。不过，这样做违反了最初的 Bell-Lapadula 模型，使实际系统从理论上讲是不安全的。

8.3 数据机密性

保护数据的机密性是一个相对古老的话题。自信息安全的研究开始，人们首先关注的就是数据机密性，即保证数据不被没有得到授权的用户非法的浏览、复制或访问。人们通常采用机密算法保护数据的机密性。对加密算法最抽象的表示为：

$$Y = f(X)与X = g(Y)$$

其中，X 为明文，Y 为密文，f 为加密算法，g 为解密算法。在信息传输之前，信息制造者首先使用加密算法 f 对明文 X 进行变换得到密文 Y，然后将 Y 分发出去。只有得到授权的人才能够知道解密算法 g，他将使用解密算法对密文 Y 进行还原，得到信息 X。随着技术的发展，人们在上面的式子中又增加了两个元素：加密密钥 p 和解密密钥 q。

$$Y = f(X,p)与X = g(Y,q)$$

现代密码学中可分为两大类，第一类是 $p = q$，也就是对称密钥加密；第二类是 $p \neq q$，也就是非对称密钥加密。这两种加密算法各有优缺点，将在下面分别介绍。

8.3.1 对称密钥加密

对称密钥加密算法也被称为私钥系统、专用密钥加密，即发送方和接收方事先确定一个密钥，这个密钥不但用来加密明文也用来对加密后的密文进行解密。在私钥体系中，最经典的算法莫过于 DES 算法了，DES 算法具有可以证明的安全性和较高的加密速度。

除了 DES 外，RC2、RC5 也是强度很高的对称密钥加密算法。

随着计算机的飞速发展，DES 算法的安全性和效率都受到了挑战。1998 年美国 EFF 使用一台专用解密机使用五十多个小时就破译了 56bit 的 DES 密钥。为此，在 1997 年，美国就开始了高级加密标准（Advanced Encryption Standard，AES）的征集活动。对 AES 算法的基本要求是：至少比三重 DES 快，至少和三重 DES 一样安全，数据分组长度为 128 位，密钥长度为 128/192/256 位。经过多次讨论之后，在 2000 年最终确定使用 Rijndael 算法作为 AES。

Rijndael 算法是由比利时人 Joan Daemen 和 Vincent Rijmen 设计的，该算法的原型是 Square 算法。这个加密体系是一种分组加密方法，信息的内容是以 128 位长度的分组作为加密基本单元的，加密密匙长度有 128，192 或 256 位多种选择。

Rijndael 算法使用的是位运算，因此即使纯粹使用软件实现，AES 也是很快的。同时，AES 可以给出算法的最佳差分特征的概率和最佳线性逼近的偏差的界，并由此分析出 AES 抵抗差分密码分析和线性密码分析的能力。

虽然需要经过复杂的变化，但对称加密算法仍具有很高的效率，这是对称加密技术仍在使用的主要原因。除了效率很高以外，在确保密钥保密的前提下，DES 和 AES 的保密强度都可以使用数学手段得到证明，他们都被证明有极高的安全性。

但是对称密钥的加密和解密采用了同一个密钥，这使得对称密钥算法具有很大的局限性，对于分布式的 Internet 环境显得有些不适合。

对称密钥算法面临的首要问题就是密钥的管理问题。密钥的分发可以采用两种方式，或者直接交送可能是素未谋面的解密方，或者是交送给一个共同的信息中心处理。对于第一种方式而言，首先需要确定对方身份的真实性，其次还要保证交送过程中的机密性。尤其是第二点，本来加密的目的之一就是解决信息传输过程中的机密性问题，现在反过来还要以数据机密性作为前提，陷入了逻辑的循环。对于第二种方式，这个信息中心不但会成为数据交换的瓶颈，同时也会成为黑客重点轰炸的目标，除此之外，如何确保信息中心的可信也是一个难以解决的问题。

对称密钥算法面临的第二个问题就是不能够为不可否认性提供保障。在第 8.1.2 节中已经介绍了不可否认性在电子商务中的巨大意义，而对称密钥算法无法提供完善的解决方案。这是对称密钥的先天不足。

对称密钥算法防外不防内。尤其是信息中心的内部人员，由于能够掌握全部的密钥，如果他们想了解机密信息的内容，可以说是易如反掌。

8.3.2　非对称密钥加密

在非对称加密体系中，密钥被分解为一对，即一把加密密钥和一把解密密钥。加密密钥为公开密钥，解密密钥为专用密钥。专用密钥由密钥对生成方掌握，公开密钥可广泛发布，它只对应于该密钥对中的专用密钥。信息交换双方利用该方案实现机密信息交

换的基本过程是：信息接收方生成一对密钥并将其中的一把作为公开密钥并公开，得到该公开密钥的信息发送方使用该密钥对机密信息进行加密后发送给信息接收方，信息接收方再用自己保存的另一把专用密钥对加密后的信息进行解密。

非对称密钥加密有着比对称密钥更广泛的用途，它不但可以用于机密信息的加密，还可以用来实现电子签名。进行电子签名的过程同保护机密信息正好相反，签名方将解密密钥公布出去，使用加密密钥对自己制造的信息加密，信息接收方使用解密密钥进行解密并阅读信息，由于加密密钥只有信息制造者知晓，所以能够采用某一解密密钥解密的信息就一定是与该解密密钥相对应的加密密钥拥有者制造的。

几乎所有的非对称密钥加密算法都是基于大整数素因子分解的困难性所研制的。对于两个大素数 p 和 q 而言，得到 $p \times q$ 的积是容易的，但将这个积再分解为 p 和 q 会非常困难。由于计算量非常大（目前，人们已经成功的分解了 512 位的整数，所以普遍认为使用 1024 位的密钥才是安全的），非对称密钥加密算法性能远远不如对称密钥加密，但由于它能够确保在不可信的信道中安全的传输信息，所以得到了广泛的应用。

在公钥加密体系中，最著名的是 RSA 算法。作为最早出现的公钥加密算法，RSA 无法从数学上证明它的安全性，但是 RSA 一直被认为是最好的公钥加密算法之一。目前，由于椭圆加密算法的良好性能，人们对椭圆加密算法寄予了厚望。

RSA 算法由 Rivest、Shamir 和 Adleman 于 1977 年提出，是最具代表性的公钥算法。RSA 算法采用如下步骤进行加密解密。

首先选择两个大素数 p 和 q，并计算 $n = p \times q$，$\varphi(n) = (p-1)(q-1)$。选择一整数 e，满足：

$$1 < e < \varphi(n), \qquad \gcd(\varphi(n), e) = 1$$

则 $n = p \times q$ 为公开密钥。私密密钥为 $d * e = 1 \bmod \varphi(n)$。对于明文 m，密文 c，则加密为 $c=m^e$（mod n），解密为 $m=c^d$（mod n）。

8.3.3 门限密码学

曾经有一个故事，说 5 个人一起出去经商，挣了一笔钱。为了路上安全，5 个人把这笔钱锁到了一个箱子里面，但又怕其他的人偷偷打开箱子独吞了财产，所以每个人都买了一把锁。但又考虑到万一出现什么紧急情况，5 个人不能同时到场还需要把钱取出来，所以商定，任意三个人同时开锁就可以把箱子打开。但是，如何排列 5 把锁锁住这个箱子，就愁煞了这 5 个人。

如果在今天，门限密码学就可以解决这样的问题。门限密码技术将一部分密码的功能分散给多人，而只有一定数量的成员合作方可完成密码运算。门限密码技术将系统的秘密 S 分解为个 n 部分 $S_1,S_2,\cdots, S_n$，系统的 N 个成员 $P_1,P_2,\cdots, P_n$ 分别拥有各自的部分秘密，任何少于 T 个成员都无法从他们的部分秘密得到任何关于系统秘密 S 的信息；而借助有效的算法，任意 T 个成员可从相应的部分秘密得到系统的秘密 S。这就是所谓的$(T–N)$

门限密码分享系统。

目前，已经出现了多种门限密码学算法，本节仅介绍基于几何超平面的门限密码算法。设：系统中有 N 个成员 $P_1, P_2,\cdots, P_n$，他们分别拥有一个 $GF(r)$上 T-维仿射空间 $AG(T, r)$的超平面

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{iT}x_T = k$$

这里 r 是大素数，法向量（$a_{i1},a_{i2},\cdots, a_{iT}$）是公开参数，$k_i$ 是每个成员的个人秘密。假设这些超平面中的任意 T 个恰好相交于唯一点 $S=(S_1,S_2,\cdots,S_n)$，作为系统的秘密。对应任意 T 个成员 $P_1,P_2,\cdots, P_n$，有下面的方程：

$$\begin{bmatrix} a_{i_1 1} & a_{i_1 2} & \cdots & a_{i_1 T} \\ a_{i_2 1} & a_{i_2 2} & \cdots & a_{i_2 T} \\ \vdots & \vdots & \vdots & \vdots \\ a_{i_T 1} & a_{i_T 2} & \cdots & a_{i_T T} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_T \end{bmatrix} = \begin{bmatrix} k_{t_1} \\ k_{t_2} \\ \vdots \\ k_{t_T} \end{bmatrix}$$

将方程写为 $AS^T=K^T$，因为方程有唯一解，故 A 可逆，即 $S^T=A^{-1}K^T$。即任何 T 个成员都可以从他们的部分秘密得到系统秘密 S，而少于 T 个成员合作则不能得到 S。

8.3.4　公开密钥基础设施

使用公开密钥密码需要一些技术的保障。例如，个人用户应该有能力保护自己的私有密码，不能泄露给任何其他人；当需要知道对方的公开密钥时，网络应该提供一种手段让用户得到对方的公开密钥，应该有技术方法确保所得到的公开密钥就是对方的，否则一旦错误地使用攻击者的公开密钥，则相当于所有的通信和交易就是与攻击者进行。同时，查询、使用和验证也应该有一个一致的标准，在密钥生成，发布和使用时也需要有一个合理的规范以保证大规模网络下的互操作性能。所有这些是实施公开密钥密码技术过程中必须解决的技术问题。这些技术问题不能解决，公开密钥密码仍旧不能投入大规模使用。

1．PKI 的概念

公开密钥密码的使用需要一个正确的方法，需要一个标准体系的支持。而以公开密钥密码的原理为基础设计和建设的，提供电子交易安全服务的一种网络应用的基础设施就是公开密钥基础设施（Public Key Infrastructure，PKI）。

早在 20 世纪 80 年代，美国学者就提出了 PKI 的概念。希望将安全和应用分离，利用公开密钥的算法和原理，建设一种普遍适用的安全基础设施，通过一个标准的统一接口，为其他应用提供安全服务。这样的构想符合工业发展的趋势，具有很好的可操作性。PKI 的倡议得到了全球的支持。

PKI 中引进一个可信任的第三方。这个第三方并非密钥管理中心，它不需要知道交易的秘密，只要交易双方共同认可就可以，每方都拥有自己的公开密钥和私有密钥。一

般认为，为实现这种第三方信任，PKI 包括 4 个方面：

（1）数字证书（certificate），数字证书将个人和他的公开密钥联系在一起。

（2）证书签发机构（Certification Authority，CA），证书签发机构产生证书并保证证书的有效性。

（3）登记机构（Registration Authority），登记机构负责验证用户的标识和身份以便产生正确的证书。

（4）证书验证路径（Certification Path），通过多层证书路径可以扩展信任范围。

另外，证书策略（Certificate Policy）是保证正确使用证书的关键，还需要有相应的设备或系统（例如，电子目录），供人们查找证书。当然，PKI 的概念不仅仅局限于以上几个独立的部分，其目标是建立一种网络上的基础设施，为网络上的电子商务或电子政务提供全面的安全服务。而且，这些服务只需要通过一个标准的接口来享用，而不需关注安全功能的实现细节。

2．PKI 提供的服务

PKI 提供的服务包括两个层次，一个为 PKI 提供的核心服务，或称为基本服务。另一个为 PKI 支撑的安全服务，属于简单的 PKI 应用所能提供的。

PKI 提供的核心服务包括认证、完整性、密钥管理、简单机密性和不可否认。这几项核心服务囊括了信息安全中的 4 个重要的要求，这就是真实性，完整性、保密性和不可否认性。

（1）认证是 PKI 提供的最基本的服务之一，这种服务可以在未曾谋面的双方之间确保认证的真实性。在目前的数学水平下可以论定，PKI 提供的认证手段是非常安全的。而这种认证方式特别适合于大规模网络和大规模的用户群。其安全性大大超过了基于口令的、基于动态口令的和基于生物特征的认证方式。在大规模网络下，这种安全优势尤为突出。

（2）PKI 提供的完整性可以通过数字签名来完成，而这种完整性还提供了对称密码方法等不能提供的不可否认保障。

（3）PKI 利用非对称的算法，提供密钥协商能力。同时，PKI 利用证书机构等提供密钥管理和简单的加密服务。PKI 在密钥管理中的一个优势是能够以合理和简单的方式实现密钥恢复，从而保证企业或国家的利益，防止密码技术被用于危害社会的行为。

（4）PKI 支撑的服务包括由加密设备提供的更强、更快的加密服务，在这种加密服务中，利用 PKI 提供的密钥交换和密钥恢复服务。例如，许多 VPN 设备就在 PKI 的基础上提供加密服务。由于 PKI 的特点，这种加密服务可以在两个未曾相识的人之间进行。

PKI 技术的广泛应用能够满足人们对网络交易安全保障的需求。当前，使用 PKI 技术的典型应用实例很多，其中，PKI 在安全电子邮件、Web、VPN 中的应用都是已经成熟并得到普及的应用。

8.4　数据完整性

数据完整性要求禁止未经授权的数据篡改，它与数据机密性紧密相连，也是更进一步的信息安全目标。保护数据完整性的方法有很多：可以通过访问控制的方法阻止数据未经授权的篡改；可以通过时间戳来判断数据最后一次修改的时间；可以通过数据校验或消息摘要的方法来判断数据是否已经被篡改。Biba 修改了 Bell-Lapadula 模型，针对数据完整性建立了 Biba 数据完整性模型。本节首先简要介绍 Biba 模型，然后详细介绍报文摘要算法。

8.4.1　Biba 完整性模型

在 Bell-Lapadula 模型的基础上，Biba 于 1977 年提出了 Biba 完整性模型。Biba 模型规定：

（1）如果主体的安全级别高于或等于客体的安全级别，那么主体可以写访问该客体。

（2）如果一个主体仅可以读访问一个客体 o，则对于另外一个客体 p 来说，如果 p 的安全级别低于或等于 o 的级别，那么该主体对于 p 可以有写访问的权限。

可以概括 Biba 模型为：

（1）读访问策略与 Bell-Lapadula 模型相同。

（2）主体不能更改安全级别比他高的客体，这也就保证了高级别客体的完整性。

Biba 模型与 Bell-Lapadula 模型一样，虽然都可以从理论上证明其安全性，但在实际应用中很难完全满足模型的要求。所以，Biba 模型也仅仅是一个理论上的模型。

8.4.2　杂凑函数与消息摘要

杂凑函数（Hash 函数）H 是一个公开的函数，杂凑函数可以将任意长的消息 M 映射为较短的、固定长度的一个值 $H(M)$，称 $H(M)$为杂凑码或消息摘要。因为杂凑码是消息中所有比特（位）的函数，所以杂凑函数可以提供错误检测的能力，改变消息中任何一个比特或几个比特都会使杂凑码发生改变。这种错误检测的能力可以判断数据是否被篡改，从而保护数据的完整性。

使用杂凑函数提供消息认证有 5 种基本方法：

（1）消息与杂凑码链接后使用对称加密算法加密，所用密钥仅为收发双方共享。这种方法可以确保接收的消息来源于消息发送方，且没有被篡改。同时，这种方法还能够提供数据机密性的保护。

（2）用对称加密算法仅对杂凑码加密。这种方法同（1）类似，有更快的速度，但不提供数据机密性的保护。

（3）采用非对称加密算法，使用发送方的私钥对杂凑码进行加密。由于只有发送方

能够产生这样的消息，所以，这种方法可以提供数字签名的功能。

（4）消息发送方同时使用对称加密和非对称加密两种算法，在（3）的基础上使用对称密钥加密消息和使用私钥变换过的杂凑码。这种方法同时提供数字签名和数据加密的功能。

（5）首先，消息收发双方共享一特定秘密 *S*，消息发送方将消息 *M* 和秘密 *S* 链接在一起并产生杂凑值，发送消息时仅发送 *M* 和杂凑值。由于发送过程中不含有 *S*，即使该消息被截获，由于杂凑值并非仅仅由 *M* 产生，而消息截获者不知道 *S* 的内容，所以仍然无法篡改消息。

可以通过杂凑算法来判断数据是否已经被篡改。对于设计巧妙的杂凑函数可以保证：

（1）不同的数据几乎不可能具有相同的摘要。

（2）通过消息摘要难以猜测消息本身。可以通过杂凑函数为数据打上标记，如果有人篡改了数据，消息摘要就会变化，新的消息摘要会与改前生成的摘要不同。通过比较摘要是否发生变化，就可以判断数据是否已经被篡改。

消息摘要（Message Digest，MD）算法是一类常用的杂凑算法，其中最常用的是MD5。MD5 算法将对输入的任意长度的信息进行计算，产生一个 128 位长度的“指纹”或“消息摘要”，假定两个不同的文件产生相同的消息摘要，或由给定的消息摘要产生原始信息在计算上是行不通的。MD5 算法能在 32 位机器上以很快的速度运行而不需要任何大型的置换列表。此算法编码很简洁。MD5 算法是 MD4 报文摘要算法的扩展，稍慢于 MD4 算法，但是，在设计上比 MD4 算法更加“保守”。设计 MD5 是因为 MD4 算法被采用的速度太快，以至于还无法证明它的正确性。MD5 后退了一步，它舍弃了一些速度以求更好的安全性。

假设有一个 *b* 位长度的输入信号，希望产生它的消息摘要，此处，*b* 不一定必须是 8 的整数倍，它可能是任意大的长度。假设设想信号的比特流如下：$m_0,m_1,\cdots,m_{b-1}$。通过下面的 5 步计算，就可以得出信号的消息摘要。

（1）补位。MD5 算法是对输入的数据进行补位，使得数据位长度 *b* 对 512 求余的结果是 448。即数据扩展至 $k\times512+448$ 位。即 $k\times64+56$ 个字节，其中 *k* 为整数。补位操作始终要执行，即使 *b* 对 512 求余的结果已是 448（此时，需要补 512 位）。具体补位操作为：补一个 1，然后补 0，直到满足上述要求。总共最少要补 1 位，最多补 512 位。

（2）补数据长度。用一个 64 位的数字表示数据的原始长度 *b*，把 *b* 用两个 32 位数表示，那么只取 *b* 的低 64 位。当遇到 *b* 大于 2^{64} 这种极少遇到的情况时，数据就被填补成长度为 512 位的倍数。也就是说，此时的数据长度是 16 个字（每个字 32 位）的整数倍数。用 $M[0\ \cdots\ N-1]$表示此时的数据，其中的 *N* 是 16 的倍数。

（3）初始化 MD 缓冲器。用一个 4 个字的缓冲器（A，B，C，D）来计算报文摘要，A，B，C，D 分别是 32 位的寄存器，初始化使用的是十六进制表示的数字：A=0X01234567，

B=0X89abcdef，C=0Xfedcba98，D=0X76543210。

（4）迭代变换函数。使用 A，B，C，D 4 个缓冲器，完成补位的原始数据和一个包含 64 个元素的常数组 T[1 ... 64]进行迭代变换。其中 $T[i]$由正弦函数构成，它的值等于经过 4294967296 次 abs(sin(i))后的值的整数部分（其中 i 是弧度）。$T[i]$为 32 位整数，用十六进制表示。MD5 总共需要进行 4 轮，每轮 16 次的变换，这里省略具体的迭代过程，感兴趣的读者可以参考相关的文档。

（5）输出结果。报文摘要产生后的形式为 A，B，C，D，也就是低位字节 A 开始，高位字节 D 结束。

MD5 算法实现很容易，据推测，要实现两个不同的消息产生相同的摘要，需要 2^{64} 次的操作；要恢复给定摘要的消息，则需要 2^{128} 次操作。

8.5　通信与网络的安全性

在网络环境下，可能危及信息安全的因素包括以下一些。

（1）伪装欺骗：指非法用户假冒合法用户身份获取敏感信息的行为。

（2）非法存取：窃取、篡改或破坏网络中存储的信息。

（3）否认抵赖：指通信方事后否认曾参与某次活动的行为。

（4）服务拒绝：指合法用户的正当申请被拒绝、更改或延迟等。

（5）截取破译：指以不正当手段获取口令或密钥，或者对加密信息与加密机制进行分析破译或篡改等。

危及路由安全方面的因素有以下一些。

（1）侦听窃密：在通信信道中窃听传输的信息。

（2）指定路由：非法用户精心设计一些绕过安全机制的路由或错误引导信息送向不正常的路由。

（3）身份攻击：指用户的身份或口令在通信时被非法截取。

（4）截取破坏：非法截取通信过程中的数据或者伪造、篡改与破坏信道中传输的信息。

（5）中继攻击：非法用户截取信息后延迟时间再发送。

（6）阻塞通信：利用网上的软硬件漏洞，发送大量无效信息使网络过载，或者设法使一些路径或网上关键互联设备瘫痪。

8.5.1　网络安全层次模型

ISO 在开放系统互联标准中定义了 7 个层次的网络参考模型，即物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。不同的网络层次之间的功能不同，针对不同层次的安全措施也是不同的。可以说，没有哪个单独的层次能够提供全部的网络

安全服务，每个层次都各尽所能。

在物理层，可以在通信线路上采用某些技术使得搭线偷听变得不可能或者容易被检测出来。在数据链路层，点对点的链路可以采用通信保密机制进行加密和解密，当信息离开一台机器时进行加密，而进入另外一台机器时进行解密。所有的细节可以全部由底层硬件实现，高层根本无法察觉。但是这种方案无法适应需要，因为在经过的每个路由器上都需要进行加密和解密，这些路由器上可能出现潜在的安全隐患，而开放网络环境并不能确定每个路由器都是安全的。虽然链路加密无论何时都很容易而且有效，但在Internet 环境中并非完全适用。

在网络层，防火墙技术可以确定来自哪些地址的信息可以或者禁止访问哪些目的地址的主机。在传输层，这个连接可以被端到端的加密，即进程到进程间的加密。应用层的安全主要是指针对用户身份进行认证并且建立起安全的通信信道。很多针对具体应用的安全方案能够有效地解决诸如电子邮件、HTTP 等特定应用的安全问题，能够提供包括身份认证、不可否认、数据加密、数据完整性检查乃至访问控制等功能，但在应用层没有一个统一的安全方案。通用安全服务应用程序接口（Generic Security Services Application Program Interface，GSS-API）的出现试图将安全服务进行抽象，为上层应用提供通用接口。在 GSS-API 接口下可以采用各种不同的安全机制来实现这些服务。

8.5.2 通信与网络安全技术

本节简单地介绍通信与网络的信息安全技术，包括 IP 层安全性、传输层安全性、应用层安全性、访问控制与目录管理、身份验证与鉴别和 Kerberos 协议等。

1. IP 层安全性

在 TCP/IP 网络中，IP 层实现的安全技术通常包括 IP 过滤技术和 IP 加密传输信道技术。其中 IP 过滤技术被路由器和防火墙产品所广泛采用，是最常见的 Internet 安全技术。在 IP 加密传输信道技术方面，互联网工程任务组（Internet Engineering Task Force，IETF）指定 IPSec 制订 IP 安全协议（IP Security Protocol，IPSP）和对应的 Internet 密钥管理协议（Internet Key Management Protocol，IKMP）的标准。IPSP 的主要目的是使需要安全服务的用户能够使用相应的加密安全体制。该体制应该是与算法独立的，可以自己选择和替换加密算法而不会对应用和上层协议产生任何影响。此外，该体制必须能支持多种安全政策，并且对其他不使用该体制的用户完全透明，IPSec 技术能够在两个网络节点间建立透明的安全加密信道。现在一些防火墙产品使用认证头部（Authentication Header，AH）或安全内容封装（Encapsulating Security Payload，ESP）支持 IPSEC 来实现 IP 层的加密，一些主要路由器厂商也声称支持 IPSec。

IP 层安全性的主要优点是它的透明性，也就是说，安全服务的提供不需要应用程序、其他通信层次和网络部件做任何改动。它的最主要的缺点是：IP 层一般对属于不同进程的包不作区别。对所有去往同一地址的包，它将按照同样的加密密钥和访问控制策略来

处理，这导致性能下降。针对面向主机的密钥分配的这些问题，RFC 1825 推荐使用面向用户的密钥分配，其中不同的连接会得到不同的加密密钥。但是，面向用户的密钥分配需要对相应的操作系统内核作比较大的改动。

IP 层非常适合提供基于主机的安全服务。相应的安全协议可以用来在 Internet 上建立安全的 IP 通道和虚拟私有网。例如，利用它对 IP 包的加密和解密功能，可以简捷地强化防火墙系统的防卫能力。

2．传输层安全性

最常见的传输层安全技术有安全套接层（Secure Socket Layer，SSL）、SOCKS（防火墙安全会话转换协议）和安全远程过程调用（Remote Procedure Call，RPC）等，其中以 SSL 的应用最为广泛。

SSL 协议分为两层，上面是 SSL 协商层，双方通过协商层约定有关加密的算法、进行身份认证等；下面是 SSL 记录层，它把上层的数据经分段、压缩后加密，由传输层传送出去。SSL 采用公钥方式进行身份认证，但是，大量数据传输仍使用对称密钥方式。通过双方协商，SSL 可以支持多种身份认证、加密和检验算法。它提供如下三类基本的安全服务。

（1）信息加密服务：在客户端与服务器之间所有业务都使用握手协议、协商密钥与算法进行加密。

（2）信息完整性服务：采用机密共享与 Hash 函数组确保 SSL 业务可正常全部到达目的地。

（3）相互认证服务：采用公开密钥技术编码标识符，使客户端与服务器在 SSL 握手时交换各自标识符实现相互认证。

传输层安全机制的主要缺点就是对应用层不透明，应用程序必须修改以使用 SSL 应用接口，而且要对传输层建立起安全机制来。同时公钥体系存在的不方便性 SSL 也同样存在，例如用户很难记住自己的公钥和私钥，必须依靠某些物理设备如 IC 卡或者磁盘来存储，这样对用户终端就有一定要求。再有就是服务方和客户方必须依赖一个证书授权中心 CA 来签发证书，双方都必须将 CA 的公钥存放在本地。为了保持 Internet 上的通用性，目前一般的 SSL 协议实现只要求服务器方向客户方出示证书以证明自己的身份而不要求用户方同样出示证书。在建立起 SSL 信道后再加密传输用户和口令实现客户方的身份认证。

与网络层安全机制相比，传输层安全机制的主要优点是它提供基于进程对进程的（而不是主机对主机的）安全服务和加密传输信道，利用公钥体系进行身份认证，安全强度高，支持用户选择的加密算法。这一成就如果再加上应用级的安全服务，就可以提供更加安全可靠的安全性能了。

3．应用层安全性

IP 层和网络层（传输层）的安全协议都无法根据所传送内容不同的安全要求作出区

别对待。如果确实想要区分一个个具体文件的不同的安全性要求，就必须在应用层采用安全机制。本质上，这意味着真正的（或许再加上机密的）数据通道还是建立在主机（或进程）之间，但却不可能区分在同一通道上传输的一个个具体文件的安全性要求。比如说，如果一个主机与另一个主机之间建立起一条安全的 IP 通道，那么两个进程间传输的所有报文都要自动地被加密。提供应用层的安全服务实际上是最灵活的处理单个文件安全性的手段。例如，一个电子邮件系统可能需要对要发出的信件的个别段落实施数据签名。较低层的协议提供的安全功能一般不会知道任何要发出的信件的段落结构，从而不可能知道该对哪一部分进行签名。只有应用层是唯一能够提供这种安全服务的层次。

一些重要的 TCP/IP 应用是对每个应用（及应用协议）分别进行修改和扩展，加入新的安全功能。例如，在 RFC 1421 至 1424 中，IETF 规定了专用强化邮件（Privacy Enbanced Mail，PEM）来为基于简单邮件传输协议（Simple Mail Transfer Protocol，SMTP）的电子邮件系统提供安全服务。

S-HTTP 是 Web 上使用的 HTTP 的安全增强版本，提供了文件级的安全机制，因此每个文件都可以被设成私人/签字状态。用作加密及签名的算法可以由参与通信的收发双方协商。S-HTTP 提供了对多种单向 Hash 函数的支持（例如，MD2、MD5 和 SHA）、对多种私钥体制的支持（例如，DES、三重 DES、RC2、RC4 和 CDMF）、对数字签名体制的支持（例如，RSA 和 DSS）。S-HTTP 和 SSL 是从不同角度提供 Web 的安全性的。S-HTTP 对单个文件作“保密/签字”之区分，而 SSL 则把参与通信的相应过程之间的数据通道按“保密”和“已认证”进行监管。

用于电子商务的安全电子交易（Secure Electronic Transaction，SET）协议规定了信用卡持卡人用其信用卡通过 Internet 进行付费的方法。这套机制的后台有一个证书颁发的基础设施，提供对 X.509 证书的支持。SET 标准在 1997 年 5 月发布了第一版，它提供数据保密、数据完整性、对于持卡人和商户的身份认证以及与其他安全系统的互操作性。

提供应用层安全的另一条思路是，通过中间件层次实现所有安全服务的功能，通过定义统一的安全服务接口向应用层提供身份认证、访问控制、数据加密等安全服务。可以将中间件层次定位为在传输层与应用层之间的独立层次，与传输层无关。SSL 也可以看成是一个独立的安全层次，但是它与 TCP/IP 紧密捆绑在一起，因此不把它看作中间件层次。

GSS-API 可以支持各种不同的加密算法、认证协议以及其他安全服务，对于用户完全透明。目前，各种安全服务都提供了 GSS-API 的接口，例如 Kerberos 就定义了自己的 GSS-API 接口。基于代理服务的中间件方案能够对应用提供更加高层的界面，甚至不需修改现有应用就能够享受中间件提供的安全服务。

4．访问控制与目录管理

资源的合法使用主要依靠网络管理员进行必要的访问控制与目录管理，用以保证网

络资源不被非法用户使用或破坏，也保证数据资源不被非法读出或改写。

根据企业的安全策略，网络管理员应当使用由网络操作系统、数据库管理系统或相应网络安全软件等提供的手段，进一步实施访问控制与目录管理方面的措施。

5. 身份验证与鉴别

除了对用户身份进行验证与鉴别，也可以对信息的真实可靠性进行验证与鉴别，用来解决冒充、抵赖、伪造或篡改等问题，除了标识用户加上口令的机制外，最常用的是数字签名技术。

数字签名可以采用秘密密钥或公开密钥技术实现，也可以使用“消息摘要”方式配合实现。签名算法可用 RSA、数字签名标准（Digital Sign Standard，DSS）等。考虑到身份验证与鉴别中可能采用相当复杂的算法或技术，目前已开始流行委托专业认证鉴别机构或公司进行身份鉴别与验证，目的是采用更先进的技术做好认证与鉴别这一关键性的安全保卫工作。

远程拨入用户认证服务（Remote Authentication Dial In User Service，RADIUS）是一种管理标准，可为分布式网络上的认证服务器提供管理服务，用来实现集中管理方式下的用户认证、口令加密、服务选择、信息过滤以及账户核对等有关任务。一个单独的 RADIUS 数据库服务器可能在多个网络上同时管理多个安全系统，其中保存有与网内所有用户有关的认证信息（如存取限制、特定的路由表、数据分组过滤规则或约定、可供核对的账户信息等），可以用来维护成千上万个用户的信息安全。

6. Kerberos 协议

Kerberos 是为分布式系统提供的认证方案，能为每种服务提供可信任的第三方认证服务。在 Kerberos 系统中只有事先在其中登记过的客户才可以申请服务。客户希望得到某一服务，需要向认证服务器申请一张能使用此服务的入场券（ticket），然后该客户把入场券传给入场券分配服务器（TGS）实施入场券的检查，在得到证实后才能使用相应的服务。其中加入了个人身份确认机制以及客户与服务器通信中的数据加密，通过系统维护数据库的方式，保存着客户、服务、权限与密钥等方面的相应信息。

8.5.3　防火墙技术

防火墙不只是一种路由器、主系统或一批向网络提供安全性的系统，而是一种运行专门的计算机安全软件（称为防火墙软件）的计算机系统，即通过软件与硬件相结合，能在企业内部网络与外部网络之间的交界处构造起一个保护层，所有的企业内部网络与外部网络之间的通信都必须经过此保护层进行检查与连接，只有授权允许的通信才能获准通过保护层。换句话说，防火墙相当于一个安全网关，能在一定意义下阻断或隔离企业内部网络与外部网络，防止非法的入侵行为或破坏行为。用防火墙可以阻止外界对内部网资源的非法访问，也可以防止内部对外部的不安全访问。

实现防火墙的产品主要有两大类：一类是网络级防火墙（主要采用包过滤技术），

另一类是应用级防火墙。现在的应用往往是把这两种技术结合起来。

1．包过滤技术

包过滤（Packet Filtering）技术是防火墙根据数据包中包头信息有选择地实施允许通过或阻断。依据防火墙内事先设定的过滤规则，检查数据流中每个数据包头部，根据数据包的源地址、目的地址、TCP/UDP 源端口号及数据包头中的各种标志位等因素来确定是否允许数据包通过，其核心是安全策略即过滤规则的设计。一般来说，不保留前后连接信息，利用包过滤技术很容易实现允许或禁止访问。相应的防火墙软件工作在传输层与网络层。

例如，基于特定的 Internet 服务的服务器驻留在特定的端口号的事实（如 TCP 端口 23 用于提供 Telnet 服务），使包过滤器可以通过规定适当的端口号来达到阻止或允许特定服务连接的目的，也可以通过规定协议号，来达到阻止或允许协议的连接，并可进一步组成一套数据包过滤规则。

包过滤技术在防火墙上的应用非常广泛，因为 CPU 用来处理包过滤的时间相对很小，而且这种防护措施对用户透明，合法用户在进出网络时，根本感觉不到它的存在，使用起来很方便。但是因为包过滤技术是在 TCP/IP 层实现的，所以包过滤的一个很大的弱点是不能在应用层级别上进行过滤，所以防卫方式比较单一，但是现在已经有一些在底层重组应用层数据的技术，从而可以对应用层数据进行检查，进而辨认一些入侵活动，收到很好的防护效果。

包过滤技术作为防火墙的应用有两类：一是路由设备在完成路由选择和数据转发之外，同时进行包过滤，这是目前较常用的方式；二是在一种称为屏蔽路由器的路由设备上启动包过滤功能。

2．应用网关技术

目前已大多采用代理服务机制，即采用一个网关来管理应用服务，在其上安装对应于每种服务的特殊代码（代理服务程序），在此网关上控制与监督各类应用层服务的网络连接。比如对外部用户（或内部用户）的 FTP、Telnet、SMTP 等服务请求，检查用户的真实身份、请求合法性和源与目的地的 IP 地址等，从而由网关决定接受或拒绝该服务请求，对于可接受的服务请求由代理服务机制连接内部网与外部网。代理服务程序的配置由企业网络管理员所控制。

目前常用的应用级防火墙大致上有三种类型，分别适合于不同规模的企业内部网：双穴主机网关、屏蔽主机网关和屏蔽子网网关。他们的共同点是需要有一台主机（称之为堡垒主机）来负责通信登记、信息转发和控制服务提供等任务。

（1）双穴主机（dual-homed）网关：由堡垒主机作为应用网关，其中装有两块网卡分别连接外部 Internet 和受保护的内部网，该主机运行防火墙软件，具有两个 IP 地址，并且能隔离内部主机与外部主机的所有可能连接。

（2）屏蔽主机（Screened Host）网关：也称甄别主机网关。在外部 Internet 与被保护

的企业内部网之间插入了堡垒主机和路由器，通常是由 IP 分组过滤路由器去过滤或甄别出可能的不安全连接，再把所有授权的应用服务连接转向应用网关的代理服务机制。

（3）屏蔽子网（Screened Subnet）网关：也称甄别子网网关，适合于较大规模的网络使用。即在外部 Internet 与被保护的企业内部网之间插入了一个独立的子网，比如在子网中有两个路由器和一台堡垒主机（其上运行防火墙软件作为应用网关），内部网与外部网的一方各有一个分组过滤路由器，可根据不同甄别规则接受或拒绝网络通信，子网中的堡垒主机（或其他可供共享的服务器资源）是外部网与内部网之间都可能访问的唯一系统。

防火墙能提供较好的安全性防护，但也带来了如下一些问题：

（1）堡垒主机或路由器本身的安全性至关重要，也可能成为影响网络性能的瓶颈。

（2）防火墙在一定程度上限制了服务的类型和用户应用的灵活性，有时会阻断一些人们常用的服务。

（3）防火墙难以防止某些内部人员的攻击，也不能防止从后门进入内部网的非法访问。对一些人们不熟悉的应用或软件（包括病毒），防火墙也缺乏防护的方法。

8.6　安全管理与安全工程

作为信息时代的资产，信息的价值随着人们对信息资源利用价值认识的不断提高而不断提升，信息的安全问题越来越受到重视。

8.6.1　安全管理的问题

各类安全技术和产品如防火墙、安全审计、入侵检测、病毒防治、数据加密、漏洞扫描、身份认证、访问控制等在计算机应用系统中逐步得到应用，并且还在不断地得到丰富和完善，但仅通过技术手段实现的安全能力是有限的，主要体现在以下三个方面：

（1）许多安全技术和产品远没有达到人们需要的水准。例如，UNIX、Windows Server 等常见的企业级操作系统，大都只达到了美国国防部 TCSEC C2 级安全认证，而且核心技术和知识产权都掌握在外国人手里，不能满足国家涉密信息系统或商业敏感信息系统的需求。再如，在计算机病毒与病毒防治软件的对抗过程中，经常是在新的计算机病毒出现并已经造成大量损失后，才开发出查杀该病毒的软件，也就是说，技术往往落后于新风险的出现。

（2）即使某些安全技术和产品在指标上达到了实际应用的某些安全需求，如果配置和管理不当，还是不能真正地实现安全需求。例如，由于风险分析欠缺、安全策略不明或是系统管理人员培训不足等原因，虽然在网络边界设置了防火墙，如果防火墙的配置出现严重漏洞，其安全功效也将会大打折扣。

（3）目前，大部分单个的网络安全管理工具比较分散，各个安全功能需要分别进行

配置，不同的管理工具之间缺乏连通性。例如，防火墙设备需要用厂商提供的专用配置管理软件进行管理，入侵检测系统要采用厂商的控制端软件实现系统状态监控，身份验证系统需要采用相应的控制中心进行管理。管理员如果要实现一个整体安全策略需要对不同的设备分别进行设置，并根据不同设备的日志和报警信息进行管理，难度较大，特别是当全局安全策略需要进行调整时，很难考虑周全和实现全局的一致性。

传统的网络管理系统所支持的协议大多以 SNMP 为主，存在下面的一些问题：

（1）很多的安全设备、安全设施并不是基于 SNMP，并不支持 SNMP 协议。

（2）多数安全设备、安全设施并不存在管理信息库（Management Information Base，MIB），或者说并不存在这样的标准。

（3）SNMP 协议本身并不适合大数据量的传输，很多安全事件、日志等安全信息不适合用 SNMP 来传送。

（4）SNMP 不支持联动和协同，而这一点随着网络的不断复杂日趋显得更加重要。

特别是最后一点，在各类安全技术与产品逐步走向成熟的同时，系统建设中不得不考虑各种安全产品、安全技术之间的协作与联动，使得安全技术与产品能够形成一个有机的整体。

这就告诉我们一个道理，即仅靠技术不能获得整体的信息安全，需要有效的安全管理来支持和补充，才能确保技术发挥其应有的安全作用，真正实现整体的信息安全，俗话说“三分技术，七分管理”，就是强调管理的重要性，在信息安全领域更是如此。于是，不少专家和厂商提出了安全的计算机网络应用体系中不可缺少的职能单元——安全管理系统。

在现有的计算机网络应用系统中，加入计算机全面安全管理职能单元的必要性在于：

（1）实现各类计算机安全技术、产品之间的协调与联动，实现有机化。

（2）充分发挥各类安全技术和产品的功能。

（3）大幅度提高整体安全能力。

（4）实现计算机安全手段与现有计算机网络应用系统的一体化。

（5）使全网安全事件能准确定位以及全网安全策略制定成为可能。

8.6.2 信息安全标准

比较著名的信息安全标准主要有 TCSEC、ITSEC、CC 和 BS7799 等。

1．TCSEC

TCSEC 将安全分为 4 个方面，即安全政策、可说明性、安全保障和文档。它把计算机系统的安全等级由低到高分为 4 类 7 级，分别是 D，C1，C2，B1，B2，B3，A1。

D 级，安全保护欠缺级。凡经检测，安全性能达不到 C1 级的划分为 D 级。D 级并非没有安全保护功能，只是太弱。

C1 级，自主安全保护级。TCB 定义和控制系统中命名用户对命名客体的访问。实施机制（例如，访问控制表）允许命名用户和（或）用户组的身份规定并控制客体的共享，阻止非授权用户读取敏感信息。

C2 级，受控存取保护级。与自主安全保护级相比，本级的 TCB 实施了粒度更细的自主访问控制，它通过登录规程、审计安全性相关事件以及隔离资源，使用户能对自己的行为负责。

B1 级，标记安全保护级。本级的 TCB 具有受控存取保护级的所有功能。此外，还需提供有关安全策略模型、数据标记以及主体对客体强制访问控制的非形式化描述；具有准确地标记输出信息的能力；消除通过测试发现的任何错误。

B2 级，结构化保护级。本级的 TCB 建立于一个明确定义的形式化安全策略模型之上，它要求将 B1 级系统中的自主和强制访问控制扩展到所有主体与客体。此外，还要考虑隐蔽通道。本级的 TCB 必须结构化为关键保护元素和非关键保护元素。TCB 的接口也必须明确定义，使其设计与实现能经受更充分的测试和更完整的复审，加强了鉴别机制；支持系统管理员和操作员的职能；提供可信设施管理；增强了配置管理控制。系统具有相当的抗渗透能力。

B3 级，安全域级。本级的 TCB 满足访问监控器需求。访问监控器是指监控主体和客体之间授权访问关系的部件。访问监控器仲裁主体对客体的全部访问。访问监控器本身是抗篡改的；必须足够小，能够分析和测试。为了满足访问监控器需求，TCB 在其构造时排除实施对安全策略来说并非必要的代码；在设计和实现时，从系统工程角度将其复杂性降低到最小程度。支持安全管理员职能；扩充审计机制，当发生与安全相关的事件时发出信号；提供系统恢复机制。系统具有很高的抗渗透能力。

A1 级，验证设计级。本级的安全功能与 B3 级相同，但最明显的不同是本级必须对相同的设计运用数字形式化证明方法加以验证，以证明安全功能的正确性。本级还规定了将安全计算机系统运送到现场安装所必须遵守的程序。

TCSEC 的一个不足是它仅仅将信息的保密性作为重点，没有重视信息的完整性和可用性。

2．ITSEC

信息技术安全评估准则（Information Technology Security Evaluation Criteria，ITSEC）是欧洲的准则，应用领域为军队、政府和商业。该标准将安全概念分为功能与评估两部分。功能准则从 F1～F10 共分 10 级。1～5 级对应于 TCSEC 的 D～A。F6～F10 级分别对应数据和程序的完整性、系统的可用性、数据通信的完整性、数据通信的保密性，以及机密性和完整性的网络安全。

与 TCSEC 不同，ITSEC 并不把保密措施直接与计算机功能相联系，而是只叙述技术安全的要求，把保密作为安全增强功能。另外，TCSEC 把保密作为安全的重点，而 ITSEC 则把完整性、可用性与保密性作为同等重要的因素。ITSEC 定义了从 E0（不满

足品质）～E6（形式化验证）级的 7 个安全等级，对于每个系统，安全功能可分别定义。

3．CC

公共准则（Common Criteria，CC）是国际标准化组织统一现有多种准则的结果，是目前最全面的评价准则。与 TCSEC 相比，CC 的优点在于：它以整个信息技术为目标，将信息技术的安全评估问题纳入一个统一的准则框架，而 TCSEC 的着眼点仅仅是安全操作系统，CC 提出了保护轮廓（Protection Profile，PP）和安全目标（Security Target，ST）等概念，采用规范的格式使安全更易于表达；CC 将安全要求区分为安全功能要求和安全保证要求，并允许按规范进行扩充。

CC 将评估过程划分为功能和保证两部分，评估等级分为 EAL1、EAL2、EAL3、EAL4、EAL5、EAL6 和 EAL7 共 7 个等级。每一级均需评估 7 个功能类，分别是配置管理、分发和操作、开发过程、指导文献、生命期的技术支持、测试和脆弱性评估。

4．BS7799

BS7799 标准是由英国标准协会制定的信息安全管理标准，是国际上具有代表性的信息安全管理体系标准，该标准包括 BS7799—1：1999《信息安全管理实施规则》和 BS7799—2：1999《信息安全管理体系规范》两部分。其中 BS7799—1：1999 于 2000 年 12 月通过 ISO 认可，正式成为国际标准，即 ISO/IEC 19799：2000《信息技术——信息安全管理实施规则》。

BS7799—2：1999 明确提出了安全控制要求，BS7799—1：1999 对应给出了通用的安全控制方法，因此可以说，BS7799—1：1999 为 BS7799—2：1999 的具体实施提供了指南。

为了更好地推进我国信息安全管理工作，公安部主持制定，国家质量技术监督局发布了《GB 17895—1999 计算机信息系统安全保护等级划分准则》，并引进了《ISO 17799：2000：信息安全管理实施准则》、《BS 7799—2：2002：信息安全管理体系实施规范》、《ISO/IEC 15408：1999（GB/T 18336：2001）——信息技术安全性评估准则》、《SSE—CMM：系统安全工程能力成熟度模型》等信息安全管理标准。信息安全标准化委员会设置了 10 个工作组，其中信息安全管理工作组负责对信息安全的行政、技术、人员等管理提出规范要求及指导指南，它包括信息安全管理指南、信息安全管理实施规范、人员培训教育及录用要求、信息安全社会化服务管理规范、信息安全保险业务规范框架和安全策略要求与指南。

8.6.3 安全管理模型

安全管理的最终目标是将系统（即管理对象）的安全风险降低到用户可接受的程度，保证系统的安全运行和使用。风险的识别与评估是安全管理的基础，风险的控制是安全管理的目的。从这个意义上讲，安全管理实际上是风险管理的过程。安全问题是动态的，是随着新的风险和系统的安全需求的不断变化而变化的。因此，安全管理应该是一个不

断改进的持续发展过程。

安全管理模型遵循管理的一般循环模式，即计划（plan）、执行（do）、检查（check）和行动（action）的持续改进模式，简称 PDCA 模式。每一次的安全管理活动循环都是在已有的安全管理策略指导下进行的，每次循环都会通过检查环节发现新的问题，然后采取行动予以改进，从而形成了安全管理策略和活动的螺旋式提升。

计算机信息系统安全管理也遵循 PDCA 持续改进模式。安全管理策略包括管理的任务、目标、对象、原则、程序和方法。安全管理活动包括制定计划、建立机构、落实政策、开展培训、检查效果和实施改进等，简要说明如下。

（1）制定计划：制定信息安全管理的具体实施、运行和维护计划。

（2）建立机构：建立相应的安全管理机构。

（3）落实措施：选择适当的安全技术和产品并实施。

（4）开展培训：对所有相关人员进行必要的安全教育和培训。

（5）检查效果：对所构建的信息安全管理体系进行综合性检查。

（6）实施改进：对检查结果进行评审，评价现有信息安全管理体系的有效性，针对存在的问题采取改进措施。

8.6.4　安全管理策略

系统的安全管理策略应包括系统安全管理的任务、目标、对象、原则、程序和方法等内容。

1．安全管理的任务

信息系统安全管理的任务是保证信息的使用和信息载体的运行安全。信息的使用安全是通过实现信息的机密性、完整性和可用性这些安全属性来保证的。信息载体包括处理载体、传输载体、存储载体和入出载体，其运行安全就是指计算系统、网络系统、存储系统和外设系统能够安全地运行。

2．安全管理的目标

信息安全管理的目标是达到信息系统所需的安全级别，将风险控制在用户可以接受的程度。

3．安全管理的对象

信息系统安全管理的对象从内涵上讲是指信息及其载体——信息系统，从外延上说其范围由实际应用环境来界定。

4．安全管理的原则

信息系统安全管理遵循如下基本原则。

（1）策略指导原则：所有的信息系统安全管理活动都应该在统一的策略指导下进行。

（2）风险评估原则：信息系统安全管理策略的制定要依据风险评估的结果。

（3）预防为主原则：在信息系统的规划、设计、采购、集成和安装中要同步考虑信

息安全问题，不可心存侥幸或事后弥补。

（4）适度安全原则：要平衡安全控制的费用与风险危害的损失，注重实效，将风险降至用户可接受的程度即可，没有必要追求绝对的、高昂代价的安全，实际上也没有绝对的安全。

（5）立足国内原则：考虑到国家安全和经济利益，安全技术和产品首先要立足国内，不能未经许可，未能消化改造直接使用境外的安全保密技术和产品设备，信息安全方面的关键技术和核心技术尤其如此。

（6）成熟技术原则：尽量选用成熟的技术，以得到可靠的安全保证。采用新技术时要慎重，要重视其成熟的程度。

（7）规范标准原则：安全系统要遵循统一的操作规范和技术标准，以保证互连通和互操作，否则，就会形成一个个安全孤岛，没有统一的整体安全可言。

（8）均衡防护原则：安全防护如同木桶装水，一是，只要木桶的木板有一块坏板，水就会从里面泄漏出来；二是，木桶中的水只和最低一块木板看齐，其他木板再高也无用。所以，安全防护措施要注意均衡性，注意是否存在薄弱环节或漏洞。

（9）分权制衡原则：要害部位的管理权限不应交给一个人管理，否则，一旦出现问题将全线崩溃。分权可以相互制约，提高安全性。

（10）全体参与原则：安全问题不只是安全管理人员的事情，全体相关人员都有责任。如果安全管理人员制定的安全制度和措施得不到相关人员的切实执行，安全隐患依然存在，安全问题就不会得到真正解决。

（11）应急恢复原则：安全管理要有应急响应预案，并且要执行必要的演练，一旦出现问题就能够马上采取应急措施，阻止风险的蔓延和恶化，将损失降低到最低程度。此外，要在灾难不能同时波及的地区设立备份中心，保持备份中心与全系统数据的一致性。一旦主系统遇到灾难而瘫痪，便可立即启动备份系统，使系统从灾难中得以恢复，保证系统的连续工作。

（12）持续发展原则：为了应对新的风险，对风险要实施动态管理。因此，要求安全系统具有延续性、可扩充性，能够持续改进，始终将风险控制在可接受的水平。

5. 安全管理的程序

信息系统安全管理的程序遵循 PDCA 循环模式的四大基本步骤如下。

（1）计划：制定工作计划，明确责任分工，安排工作进度，突出工作重点，形成工作文件。

（2）执行：按照计划展开各项工作，包括建立权威的安全机构，落实必要的安全措施，开展全员的安全培训等。

（3）检查：对上述工作所构建的信息安全管理体系进行符合性检查，包括是否符合法律法规的要求，是否符合安全管理的原则，是否符合安全技术的标准，是否符合风险控制的指标等，并报告结果。

（4）行动：依据上述检查结果，对现有信息安全管理策略的适宜性进行评审与评估，评价现有信息安全管理体系的有效性，采取改进措施。

6．安全管理的方法

安全管理根据具体管理对象的不同，采用不同的具体管理方法，信息系统安全管理的具体对象包括机构、人员、软件、设备、介质、涉密信息、技术文档、网络连接、门户网站、应急恢复、安全审计、场地设施等。

8.6.5　安全管理框架

企业安全管理（Enterprise Security Management，ESM）是目前国际上兴起的一种整体安全管理框架，它的主要思想是采用多种智能 Agent 和安全控制中心，在统一安全策略的指导下，将系统中的各个安全部件协同起来，实现总体的安全策略，并且能够在多个安全部件协同的基础上实现适时监控、报表处理、统计分析等。这样的体系架构具备适应性强（能够适用于各种网络和系统环境）、可扩充性强、集中化安全管理等优点，已成为网络安全整体解决方案的发展方向。

ESM 框架体系主要是为了解决目前各类安全产品各自为阵、难以组成一个整体安全防御体系的问题。在信息技术被大量采用的今天，安全威胁可能来自内部和外部，许多机构发现来自内部的威胁（特别是来自心怀不满的员工和临时雇员）可能造成更大的损失。但是目前大多数的安全产品只能保护某个点上的安全，例如，可以安装防病毒软件查杀病毒，安装防火墙抵御外部的普通攻击，采用入侵检测系统发现入侵等。遗憾的是，许多机构在部署了某些安全产品后，仍然无法保证网络的安全。不同的安全产品之间也缺乏联系，难以管理。真正的整体安全是在一个整体的安全策略下，安全产品和非安全产品以及管理制度相互协调的基础上才能够实现，因此目前许多国外的安全厂商都在大力推行 ESM 的体系架构。

8.6.6　安全管理系统实现的功能

安全管理系统是信息系统安全的必要组成部分，它为系统管理员和用户提供对整体安全系统的监管，在计算机网络应用体系与各类安全技术、安全产品、安全防御措施等安全手段之间搭起桥梁，使得各类安全手段能与现有的计算机网络应用体系紧密结合实现无缝连接，促成计算机网络安全与计算机网络应用的真正的一体化。

一般，安全管理系统需要实现以下的功能。

1．实现对系统中多种安全机制的统一监控和管理

根据网络中各种不同的安全系统，有针对性地进行统一的监控管理，包括监视各种安全机制的各个部件的运行状况；发现各种安全机制的运行异常情况；向各种安全机制发布相应的总体安全策略；通过安全管理系统，实现对各种安全机制的实时操控；收集各种安全机制执行安全策略的结果等。

安全管理系统对各种安全机制的监控和管理可以利用各个安全子系统中已有的信息采集和控制机制来实现，也可以采用直接与安全设备交互的方式进行，主要取决于各个安全子系统自身的构架以及提供的管理接口。

2．实现各类安全设施之间的互动和联动

安全管理系统提供自动响应功能，在网上实现对各类相关的安全设施之间的互动和联动，并对联动情况进行监控。

一个系统中的不同安全机制通常通过两种方式来实现安全设施之间的联动：

（1）利用原有的设备之间的互动功能，例如，入侵检测系统与防火墙之间的互动，以及审计系统与身份验证系统之间的互动等。安全管理系统根据原有安全系统之间能够实现的互操作功能来统一设置联动的策略，并监视这些设施之间的联动情况。

（2）安全管理系统通过从收集各个安全系统中产生的各类数据，并采用自动或手动响应引擎，根据实现设定的安全策略以及规则，对相关的安全子系统以及安全设备进行设置和操控，以实现间接的联动。

3．实现基于权限控制的统一管理和区域自治

安全管理系统提供统一的安全管理，并为不同级别和性质的管理员提供不同层次和性质的管理视图。由于大型系统通常是一个复杂的分布式大规模网络，因此运行管理中心、汇接层以及接入单位的网络管理员具有不同的职责。系统不但能够提供运行管理中心的管理员对所有安全系统宏观的管理视图，也能够为各个接入单位和区域分管的管理员对自己管辖区域内的安全设备和安全系统部件进行区域自治管理，此外，还能够通过安全管理系统对分布于整个网络的某个安全子系统进行整体安全策略的发放和状态监测及管理。

4．实现安全事件事务管理

安全管理系统根据从各种安全机制收集的资料进行安全事件的事务处理，包括对重复安全事件的合并处理；对相互关联的安全事件进行的合并处理；根据相近零碎的历史事件集合对安全事件进行确认；智能判断事件的真正起因，并提供人工修正判断的机制；根据管理员的职责将合并与确认后的事件通知响应责任人，并提供处理建议；根据责任人处理事件的情况以及结果，确定是否对事件性质进行升级等。

5．实现各类实时报警措施

系统将实现各类实施报警措施，如管理员控制台声音报警、管理员控制台界面报警、E-mail 报警、寻呼机报警、手机中文短消息报警、Yahoo Message 报警、ICQ 报警、MSN 即时信息报警等。

6．实现安全事件和数据的宏观统计分析和决策支持

系统支持对长期积累的数据进行宏观统计、分析和决策支持。宏观统计分析主要是在长期积累的大量数据的基础上，对安全事件进行综合分析，包括对安全事件的类型、

来源、目的、产生的效果、起因、发生的时段进行综合分析，得到宏观的规律；对重要事件的来源进行综合查证，定位到个人；对相近时段发生的各种事件进行相关性分析，得出各类不同事件相互联系的规律，并指导自动联动规则和安全策略的制定；根据宏观统计的结果，提供决策支持，并进行知识积累，为各类安全事件提供处理建议等。

7．支持应急响应

系统支持在紧急情况下的应急响应，包括系统设计充分考虑备份措施和应急措施，以备紧急状态下使用；系统支持制定应急情况预案，采用基于综合条件设置和响应动作设置来定义应急情况以及应急响应措施；在发生紧急情况时，能够根据事件的综合，发现系统处于严重异常状态中，并以各种措施通知责任人员；在确认所处紧急状态的前提下，启动预案中已经设定的整套响应措施，系统将自动调用各类外部系统和备份机制，实现应急预案中设定的批量操作，并对整个操作的过程进行跟踪和记录等。

8.6.7　系统安全工程

安全工程如同网络工程、信息系统工程的实施过程一样，是一个包括安全需求分析、安全风险分析、安全基线验证、安全体系设计、安全策略实施等多方面的系统工程。

由美国国家安全局推出的系统安全工程能力成熟度模型（Systems Security Engineering Capability Maturity Model，SSE-CMM）的基本思想是建立和完善一套成熟的、可度量的安全工程过程。该模型定义了一个安全工程过程应有的特征，这些特征是完善安全工程的根本保证。这个安全工程对于任何工程活动均是清晰定义的、可管理的、可测量的、可控制的，并且是有效的。SSE-CMM 模型及其评定方法汇集了业界范围内常见的实施方法，提供了一套包括政府及产业的标准度量体系，确保了在处理硬件、软件、系统和组织安全问题的工程实施活动后，能够得到一个完整意义上的安全结果。

1．SSE-CMM 模型体系结构

为了将安全工程思想变为一种有效的工程规范，在 SSE-CMM 模型中，定义了 22 个安全方面的过程域（Process Areas，PA），并将每个过程域按其能力由低到高分为 0～5 六个级别，在每个过程域中提出了要控制和达到的目标。为了实现这些目标，在每个过程域中又包括许多具体的基本实施（Basic Practice，BP）。这些基本实施规范了工作流程，是保证过程目标有效控制的重要手段。按照解决问题的不同，过程域可以分为三类：一类是工程过程域，包括 11 个过程；一类是项目过程域，包括 5 个过程；还有一类是组织过程域，包括 6 个过程。

SSE-CMM 将系统安全工程分为三类，它们是风险过程、工程过程和确认过程。风险过程是指对要实施安全工程的系统进行风险分析，分析各种可能对系统构成威胁的影响因素、系统本身的脆弱性，以及如果威胁因素起作用可能对系统造成的影响；工程过程是指工程队伍根据风险分析的结果、有关系统需求、可应用的法律法规和方针政策等

信息，与客户一起识别和定义系统的安全需要，在综合考虑包括成本、性能、技术风险和使用难易程度等各种因素和各种替代方案之后创建出解决方案，然后用该方案指导安全系统的开发和建设，并对系统进行不间断的监测，以保证风险不至于增大到不能接受的程度；确认过程是对安全工程过程和质量结果进行测试和验证，从而得出系统安全是否可信的结果。随着这三个过程的不断执行，工程队伍的过程能力也不断成熟。

2．SSE-CMM 的过程能力水平

过程能力是由一组通用实施来衡量，通用实施是对所有工程过程都通用的工程实践。按照工程队伍对通用实施的执行情况，可以将每个过程域按能力的高低分成 6 个级别，即从第 0 级到第 5 级。

0 级能力水平指非执行能力级。非执行能力级的过程没有共同特征和通用实践（GP），在开发过程中没有安全工程思想的应用。在这一级的过程水平中也能够完成一些工作，但是当工程队伍中的关键人物不在或者当工程本身变得越来越复杂时，就难以保证任务的完成。

1 级能力水平指非正常执行的能力水平。所有的基本实施在一定程度上都能被执行，因此对过程能力缺少连续的计划和跟踪。过程的完善能力仍然取决于个人的知识和努力程度。产品质量和生产效率由工程队伍的所有人员的出色工作来保证。由于过程的执行还主要靠经验，对执行结果无明确要求，所以执行活动的能力是不可重复和被其他过程所借鉴的。

2 级能力水平是指具有计划与跟踪的能力水平。由于组织的过程能力取决于安全工程基本实施的效率，因此与基本实施有关的工作过程可以被总结和控制。它与 1 级能力水平不同之处在于，此过程中的基本实施是可以重复和被其他组织借鉴的。

3 级能力水平是指完好定义的能力级水平。过程中的所有基本实施应按照完善定义的规范来进行，这些规范是工程队伍根据长期经验而总结出来的。它与 2 级能力水平不同之处在于定义了一个被接受的标准规范，基本的实施可以反映出过程的特征，过程的能力可以直接转到其他工程活动中。

4 级能力水平是定量控制级水平。对每个已定义的过程和相联系的工作都设定出可度量的过程目标，可以对工程队伍和工程的进展进行定量的预测和控制。

5 级能力水平是持续完善的能力水平。从过程能力的角度看它是最高水平，在此水平下已经建立了对过程效率的定性和定量的目标，而且可以准确度量过程持续改善所获得的效益。

SSE-CMM 模型本身并不是安全技术模型，但它给出了信息系统安全工程需考虑的关键过程域，可指导安全工程从单一的安全设备配置转向系统地解决整个工程的风险评估、安全策略形成、安全方案提出、实施和生存周期控制等问题。

本章参考文献

[1] 张世永. 网络安全原理与应用. 北京：科学出版社，2003

[2] 胡道元，闵京华. 网络安全. 北京：清华大学出版社，2004

[3] 李孟珂，余祥宣. 基于角色的访问控制技术及应用. 计算机应用研究，2000（10）：44-47

[4] 胡道元. 计算机网络（高级）. 北京：清华大学出版社，1999

第 9 章　嵌入式系统设计

嵌入式系统设计的主要任务是定义系统的功能、决定系统的架构，并将功能映射到架构。这里，系统架构既包括软件系统架构，也包括硬件系统架构。一种架构可以映射到各种不同的物理实现，每种实现表示不同的取舍。同时，还要满足某些设计指标，并使其他的设计指标也同时达到最佳化。

嵌入式系统的设计方法跟一般的硬件设计、软件开发的方法不同，是采用硬件和软件协同设计的方法，开发过程不仅涉及软件领域的知识，还涉及硬件领域的综合知识，甚至还涉及机械等方面的知识。要求设计师必须熟悉并能自如地运用这些领域的各种技术，才能使所设计的系统达到最优。

9.1　嵌入式系统概论

嵌入式系统是一个面向应用、面向产品的专用计算机系统。无论从技术还从应用上，从软件系统或硬件组成上，嵌入式系统都有许多不同于通用计算机系统的地方。习惯上，人们通常把嵌入式系统归到非通用计算机系统这一类。

9.1.1　嵌入式系统的基本概念

嵌入式系统是一种以应用为中心，以计算机技术为基础，可以适应不同应用对功能、可靠性、成本、体积、功耗等方面的要求，集可配置可裁减的软、硬件于一体的专用计算机系统。它具有很强的灵活性，主要由嵌入式硬件平台、相关支撑硬件、嵌入式操作系统、支撑软件以及应用软件组成。

嵌入式系统具有以下特点：

（1）系统专用性强。嵌入式系统是针对具体应用的专门系统。它的个性化很强，软件和硬件结合紧密。一般要针对硬件进行软件的开发和移植，根据硬件的变化和增减对软件进行修改。由于嵌入式系统总是用来完成某一特定任务的，整个系统与具体应用有机地结合在一起，升级换代也以更换整个产品的方式进行，因此，一个嵌入式产品一旦进入市场，一般具有较长的生命周期。

（2）系统实时性强。嵌入式系统有相当一部分系统对外来事件要求在限定的时间内及时做出响应，具有实时性。

（3）软、硬件依赖性强。嵌入式系统的专用性决定了其软、硬件的互相依赖性很强，两者必须协同设计，已达到共同实现预定功能的目的，并满足性能、成本和可靠性等方

面的严格要求。

（4）处理器专用。嵌入式系统的处理器与通用计算机的处理器的最大不同之处在于，嵌入式系统的处理器一般是为某一特定目的和应用而专门设计的。通常具有功耗低、体积小、集成度高等优点，能够把许多在通用计算机上需要由板卡完成的任务和功能集成到芯片内部，从而有利于嵌入式系统的小型化和移动能力的增强。

（5）多种技术紧密结合。嵌入式系统通常是计算机技术、半导体技术、电力电子技术及机械技术与各行业的具体应用相结合的产物。通用计算机技术也离不开这些技术，但它们相互结合的紧密程度不及嵌入式系统。

（6）系统透明性。嵌入式系统在形态上与通用计算机系统差异甚大。它的输入设备往往不是常见的鼠标和键盘之类的设备，甚至没有输出装置，用户可能根本感觉不到它所使用的设备中有嵌入式计算机系统的存在，即使知道也不必关心这个嵌入式计算机系统的相关情况。

（7）系统资源受限。嵌入式系统为了达到结构紧凑、可靠性高及降低系统成本的目的，其存储容量、输入/输出设备数量和处理器的处理能力都比较有限。

9.1.2　实时系统的基本概念

简单地说，实时系统可以看成对外部事件及时响应的系统。现实世界中，并非所有的嵌入式系统都具有实时特性，所有的实时系统也不一定都是嵌入式的。但这两种系统并不互相排斥，兼有这两种系统特性的系统称为实时嵌入式系统。

为了后续内容的叙述方便，下面先介绍实时系统有关的几个概念。

（1）逻辑（或功能）正确。指系统对外部事件的处理能够产生正确的结果。

（2）时间正确。指系统对外部事件的处理必须在预定的周期内完成。

（3）死线（deadline）。指系统必须对外部事件处理的最迟时间界限，错过此界限可能产生严重后果。通常，计算必须在到达死线前完成。

（4）实时系统。指功能正确和时间正确同时满足的系统，二者同等重要。换言之，实时系统有时间约束并且是死线驱动的。但是，在某些系统中，为了保证功能正确性，有可能牺牲时间正确性。

根据对错失死线的容忍程度，可以将实时系统分为软实时系统和硬实时系统。

硬实时系统是指系统必须满足其灵活性接近零死线要求的实时系统。死线必须满足，否则，就会产生灾难性后果，并且死线之后得到的处理结果或是零级无用，或是高度贬值。

软实时系统是指必须满足死线的要求，但是有一定灵活性的实时系统。死线可以包含可变的容忍等级、平均的时间死线，甚至是带有不同程度的可接受性的响应时间的统计分布。在软实时系统中，死线错失不会导致系统失败。由于一个或多个错失的死线对软实时系统的运行没有决定性的影响，软实时系统不必预测是否可能有悬而未决的死线

错失。软实时系统在探知到错失一个死线时可以启动一个恢复进程。

希赛教育专家提示：在实时系统中，任务的开始时间与死线或完成时间同样重要，由于任务缺少需要的资源，例如，CPU 和内存，就有可能阻碍任务的开始并直接导致错失任务的完成死线。因此，死线问题演变成资源的调度问题。这一点对调度算法和任务设计都有至关重要的影响。

9.2 嵌入式系统的基本架构

嵌入式系统一般都由软件和硬件两个部分组成，其中嵌入式处理器、存储器和外部设备构成整个系统的硬件基础。嵌入式系统的软件部分可以分为三个层次，分别是系统软件、支撑软件和应用软件。系统软件和支撑软件是基础，应用软件则是最能体现整个嵌入式系统的特点和功能的部分。

9.2.1 硬件架构

图 9-1 是一个嵌入式系统的基本硬件架构。

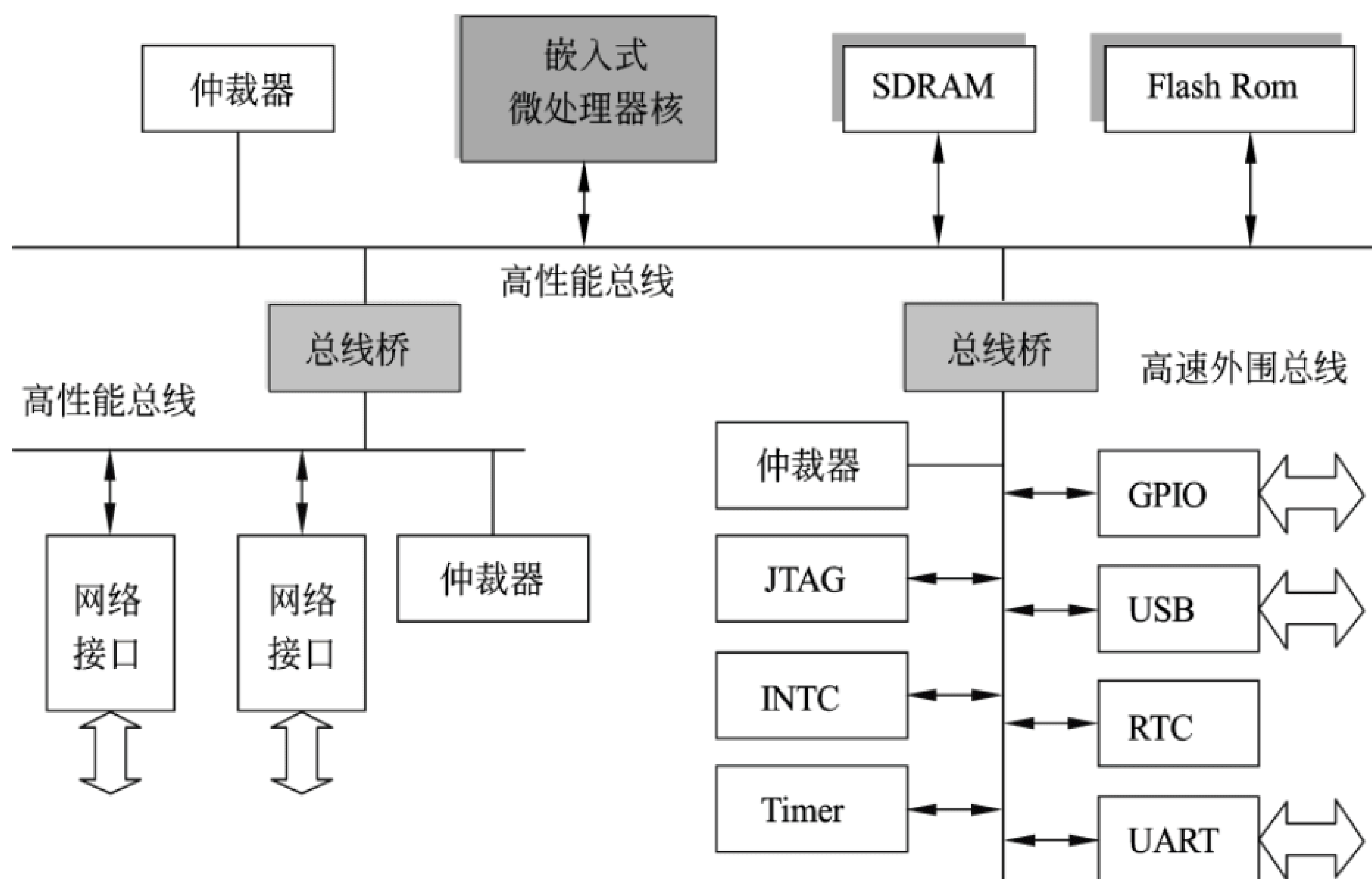


图 9-1 嵌入式硬件平台的系统架构

微处理器是整个嵌入式系统的核心，负责控制系统的执行。外部设备是嵌入式系统与外界交互的通道，常见的外部设备有 flash 存储器、键盘、输入笔、触摸屏、液晶显示器等输入/输出设备，在很多嵌入式系统中还有与系统用途紧密相关的各种专用外设。嵌入式系统中经常使用的存储器有三种类型，分别是随机存取存储器（Random Access Memory，RAM）、只读存储器（Read Only Memory，ROM）和混合存储器。系统的存

储器用于存放系统的程序代码、数据和系统运行的结果。

嵌入式系统的核心部件是各种类型的嵌入式处理器。据不完全统计，目前世界上嵌入式处理器的种类已经超过了 1000 种，比较流行的也有 30 几个系列。根据目前的使用情况，嵌入式处理器可以分为如下几类：

（1）嵌入式微处理器。嵌入式微处理器（Embedded Micro Processing Unit，EMPU）由通用计算机中的中央处理器（Center Processing Unit，CPU）演变而来。嵌入式微处理器在功能上跟普通的微处理器基本一致，但是它具有体积小、功耗低、质量轻、成本低及可靠性高的优点。通常，嵌入式微处理器和 ROM、RAM、总线接口及外设接口等部件安装在一块电路板上，称为单板计算机。目前，主要的嵌入式微处理器有 AM186/88、386EX、SC-400、POWER PC、MIPS 和 ARM 等系列。

（2）嵌入式微控制器。嵌入式微控制器（Embedded Micro Controlling Unit，EMCU）又称为单片机，就是整个计算机系统都集成到一块芯片中。嵌入式微控制器一般以某一种微处理器内核为核心，芯片内部集成有 ROM/EPROM/E2PROM、RAM、总线、总线逻辑、定时器/计数器、WatchDog（监督定时器）、并口/串口、数模/模数转换器、闪存等必要外设。与嵌入式微处理器相比，嵌入式微控制器的最大特点是单片化，因此体积更小、功耗和成本更低，可靠性更高。目前，嵌入式微控制器的品种和数量最多，约占嵌入式系统市场份额的 70%。比较有代表性的通用系列有 8051 系列、MCS-96/196/296、C166/167、MC68HC05/11/12/16 级 68300 等。还有许多半通用系列，例如，用于支持 USB（Universal Serial Bus，通用串行总线）接口的 MCU 8XC930/931、C540、C541，以及用于支持 I2C、现场总线等的各种微控制器。

（3）嵌入式数字信号处理器。嵌入式数字信号处理器（Embedded Digital Signal Processor，EDSP）是一种专门用于信号处理的处理器，DSP 芯片内部采用程序和数据分开的哈佛结构，具有专门的硬件乘法器，广泛采用流水线操作，提供特殊的 DSP 指令，可以用来快速实现各种数字信号的处理算法。嵌入式数字信号处理器主要有如下特点：

- 在一个指令周期内可以完成一次乘法和一次加法。
- 程序和数据空间分开，可以同时访问指令和数据。
- 片内具有快速存储器，通常可以通过独立的数据总线在两块芯片中同时访问。
- 具有低开销或无开销循环及跳转的硬件支持。
- 快速的中断处理和硬件 I/O 接口支持。
- 具有在单周期内操作的多个硬件地址产生器。
- 可以并行执行多个操作。
- 支持流水线操作。

目前，数字信号处理器在嵌入式系统中使用非常广泛，例如，数字滤波、快速傅立叶变换及频谱分析等。同时，嵌入式系统的智能化也是推动 EDSP 发展的一个动力，例如，各种带有智能逻辑的消费类产品、生物信息识别终端、带有加密/解密算法的设备、

实时语音压缩和解压系统以及虚拟现实显示装置等，这类系统上的智能化算法一般运算量都比较大，这恰好可以充分发挥数字信号处理器的长处。

（4）嵌入式片上系统。嵌入式片上系统（Embedded System on Chip，ESoC）是一种在一块芯片上集成很多功能模块的复杂系统。例如，微处理器内核、RAM、USB、IEEE 1394、Bluetooth（蓝牙）等。以往这些单元按照各自的功能做成一个个独立的芯片，通过电路板与其他单元组成一个系统。现在将这些本来在电路板上的单元都集成到一个芯片中，构成一个嵌入式片上系统，从而大幅度缩小了系统的体积、降低了系统的复杂度、增强了系统的可靠性。在大量生产时，生产成本也远远低于单元部件组成的电路板系统。

嵌入式片上系统可以分为通用片上系统和专用片上系统两类。通用类的主要产品有Siemens 公司的 Trocore、Motorola 的 M-Core、某些 ARM 系列的器件等。专用类的嵌入式片上系统一般是针对某一或某些系统而设计的。具有代表性的产品有 Philips 公司的 Smart XA，它将 XA 单片机的内核和支持超过 2048 位复杂 RSA 算法的 CCU 单元制作在一个芯片上，形成一个可加载 Java 或 C 的专用嵌入式片上系统，可用于网络安全等方面。

9.2.2 软件架构

随着嵌入式技术的发展，特别是在后 PC 时代，嵌入式软件系统得到了极大的丰富和发展，形成了一个完整的软件体系，如图 9-2 所示。这个体系自底向上由三部分组成，分别是嵌入式操作系统、支撑软件和应用软件。

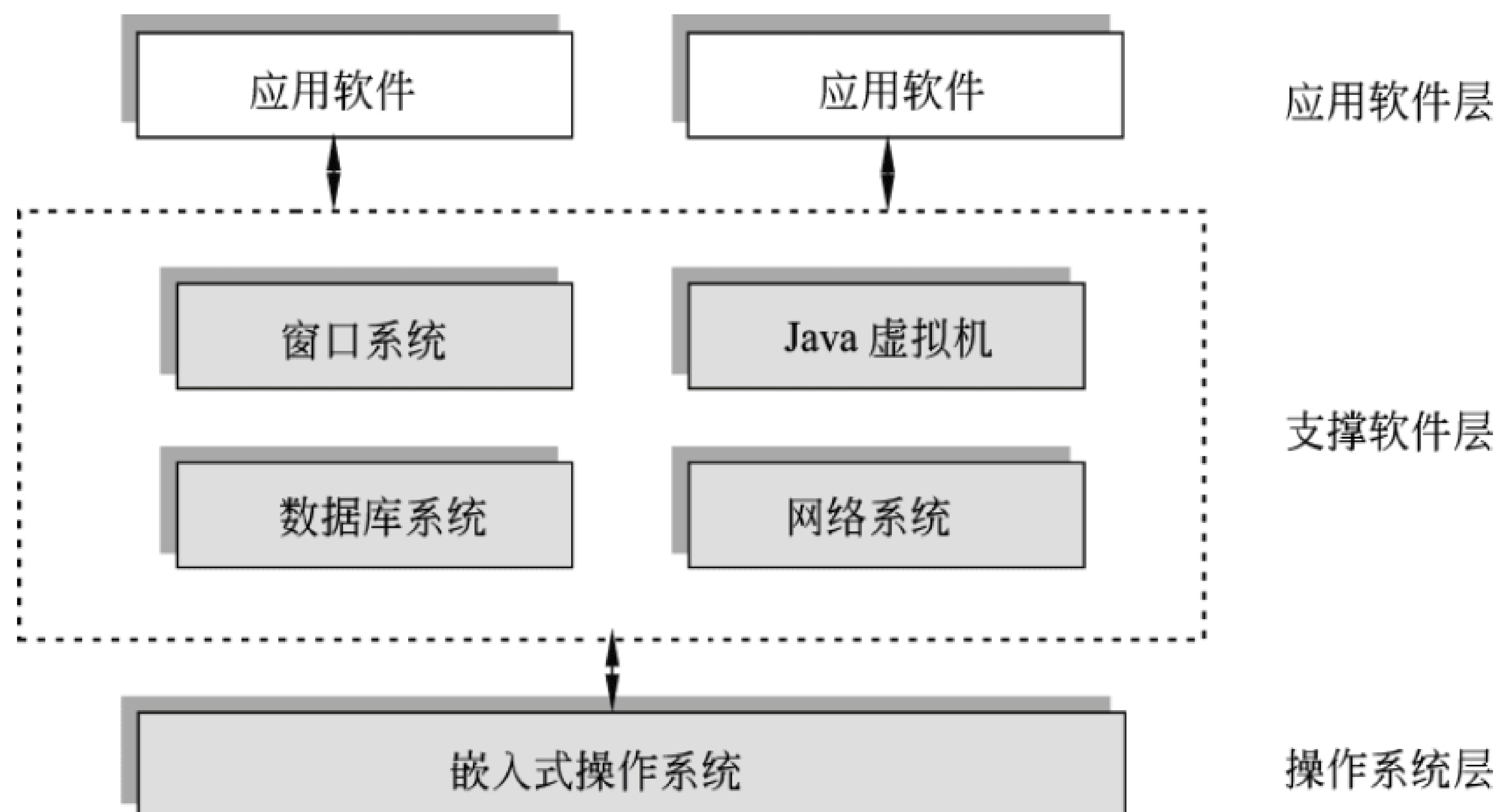


图 9-2 嵌入式系统的软件架构

1．操作系统

嵌入式操作系统由操作系统内核、应用程序接口、设备驱动程序接口等几部分组成。嵌入式操作一般采用微内核结构。操作系统只负责进程的调度、进程间的通信、内存分配及异常与中断管理最基本的任务，其他大部分的功能则由支撑软件完成。

2．支撑软件

嵌入式系统中的支撑软件由窗口系统、网络系统、数据库管理系统及 Java 虚拟机等几部分组成。对于嵌入式系统来讲，软件的开发环境大部分在通用台式计算机和工作站上运行，但从逻辑上讲，它仍然被认为是嵌入式系统支撑软件的一部分。支撑软件一般用于一些浅度嵌入的系统中，如智能手机、个人数字助理等。

3．应用软件

嵌入式系统中的应用软件是系统整体功能的集中体现。系统的能力总是通过应用软件表现出来的，一个嵌入式系统可以没有支撑软件，甚至可以没有操作系统，但不可以没有应用软件，否则它就不可能成为一个系统。从范围上讲，嵌入式系统的应用软件涉及工业控制、家电、商业、通信等诸多领域。从跟用户的交互方式上讲，有跟桌面系统类似的软件，也有潜入的程度很深，使用户感觉不到其存在的应用软件。从运行环境上讲，有在操作系统和支撑软件上运行的软件，也有直接在硬件上运行的应用软件。

9.3　嵌入式操作系统

目前，嵌入式系统应用软件的设计方案随应用领域的不同而不同，相对简单的系统一般采用裸机设计，即直接基于处理器编写软件，这类系统一般功能简单，代码量少。而对于功能复杂的系统，若也采用裸机开发的方案，其代码开发的工作量和难度都很大，系统的可靠性和可维护性势必难以保证。

随着各种类型嵌入式操作系统的成熟和发展，与通用系统的开发方法相似，基于操作系统的开发方案逐渐成为开发的主流。嵌入式操作系统作为应用软件的运行平台，已经成为许多嵌入式系统的关键。在诸多类型的嵌入式操作系统中，实时嵌入式操作系统是最具代表性的一类，它融合了几乎所有类型的嵌入式操作系统的特点，所以本节主要以实时嵌入式操作系统的特性和概念为主，对嵌入式操作系统的基本概念与特点、基本架构、内核服务、内核对象、多任务的调度等核心内容进行全面的介绍。

9.3.1　概念与特点

嵌入式操作系统是指运行在嵌入式计算机系统上支持嵌入式应用程序的操作系统，是用于控制和管理嵌入式系统中的硬件和软件资源、提供系统服务的软件集合。嵌入式操作系统是嵌入式软件的一个重要组成部分。它的出现提高了嵌入式软件开发的效率，提高了应用软件的可移植性，有力地推动了嵌入式系统的发展。

与通用操作系统相比，嵌入式操作系统主要有以下特点：

（1）微型化。嵌入式操作系统的运行平台不是通用计算机，而是嵌入式计算机系统。这类系统一般没有大容量的内存，几乎没有外存，因此，嵌入式操作系统必须做得小巧，以尽量少占用系统资源。为了提高系统地执行速度和系统的可靠性，嵌入式系统中的软

件一般都固化在存储器芯片中，而不是存放在磁盘等载体中。

（2）代码质量高。在大多数应用中，存储空间依然是宝贵的资源，这就要求程序代码的质量要高，代码要尽量精简。

（3）专业化。嵌入式系统的硬件平台多种多样，处理器的更新速度快，每种都是针对不同的应用领域而专门设计。因此，嵌入式操作系统要有很好适应性和移植性，还要支持多种开发平台。

（4）实时性强。嵌入式系统广泛应用于过程控制、数据采集、通信、多媒体信息处理等要求实时响应的场合，因此实时性成为嵌入式操作系统的又一特点。

（5）可裁减、可配置。应用的多样性要求嵌入式操作系统具有较强的适应能力，能够根据应用的特点和具体要求进行灵活配置和合理裁减，以适应微型化和专业化的要求。

9.3.2 一般结构

与通用计算机系统上的操作系统一样，嵌入式操作系统隔离了用户与计算机系统的硬件，为用户提供了功能强大的虚拟计算机系统，如图 9-3 所示。嵌入式操作系统主要由应用程序接口、设备驱动、操作系统内核等几部分组成。

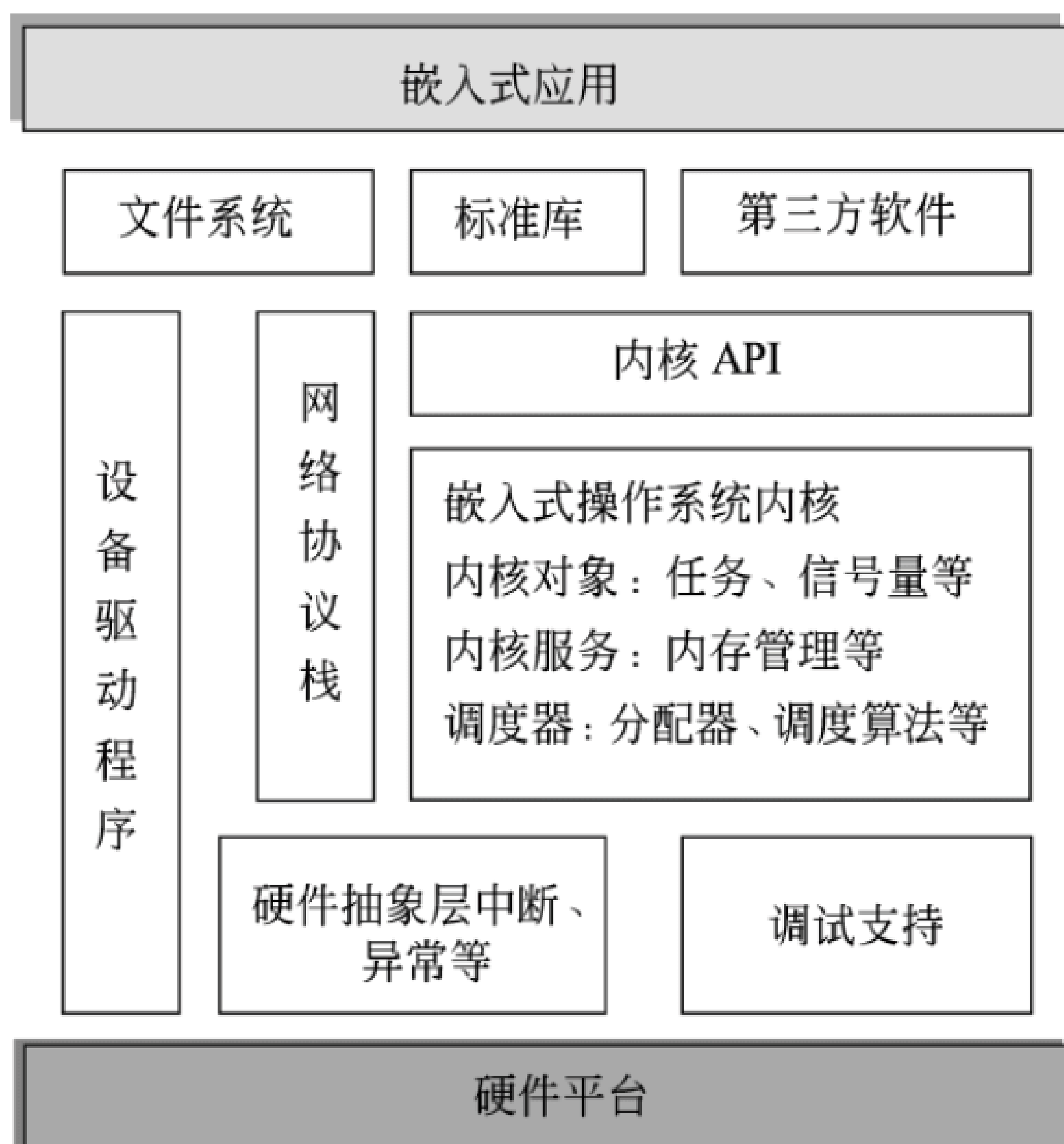


图 9-3 嵌入式操作系统的一般结构

嵌入式操作系统是一个按时序方式调度执行、管理系统资源并为应用代码提供服务的基础软件。每个嵌入式操作系统都有一个内核。另一方面，嵌入式操作系统也可以是各种模块的有机组合，包括内核、文件系统、网络协议栈和其他部件。但是，如图 9-4 所示，大多数内核都包含以下三个公共部件：

（1）调度器。是嵌入式操作系统的心脏，提供一组算法决定何时执行哪个任务。

（2）内核对象。是特殊的内核构件，帮助创建嵌入式应用。

（3）内核服务。是内核在对象上执行的操作或通用操作。

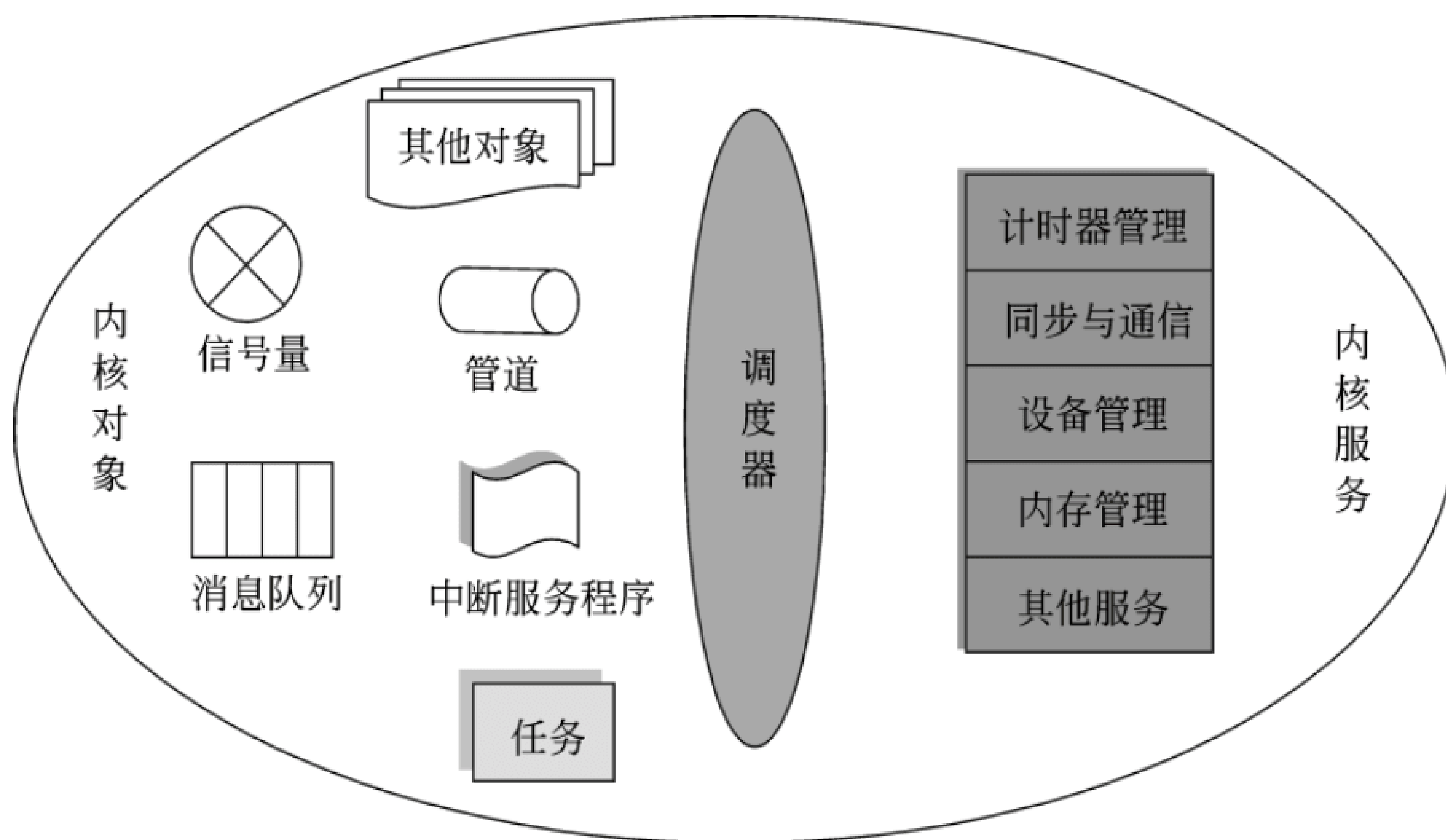


图 9-4　嵌入式操作系统的内核部件

9.3.3　多任务调度

为了描述嵌入式系统任务的调度，本节首先介绍一些相关的概念。

1．基本概念

（1）任务。是独立执行的线程，线程中包含独立的可调度的指令序列。实时应用程序的设计过程包括如何把问题分割成多个任务，每个任务是整个应用的一个组成部分，每个任务被赋予一定的优先级，有自己的一套寄存器和栈空间。在大多数典型的抢占式调度内核中，在任何时候，无论是系统任务还是应用任务，其状态都会处于：就绪、运行、阻塞三个状态之一。另外，某些商业内核还定义了挂起、延迟等颗粒更细的状态。

（2）任务对象。任务是由不同的参数集合和支持的数据结构定义。在创建任务时，每个任务都拥有一个相关的名字，一个唯一的标识号 ID，一个优先级、一个任务控制块、一个堆栈和一个任务的执行例程，这些部件一起组成一个任务对象。

（3）多任务。是操作系统在预定的死线内处理多个活动的能力。多任务的运行使 CPU 的利用率得到最大地发挥，并使应用程序模块化。随着调度的任务数量的增加，对 CPU 的性能需求也随之增加，主要是由于线程运行的上下文切换增加的缘故。

（4）调度器。调度器是每个内核的心脏，调度器提供决定何时哪个任务运行的算法。多数实时内核是基于优先级调度的。

（5）可调度实体。可调度实体是一个可以根据预定义的调度算法，竞争到系统执行

时间的内核对象。

（6）上下文切换。每个任务都有自己的上下文，它是每次被调度运行时所要求的寄存器状态，当多任务内核决定运行另外的任务时，它保存正在运行的任务的上下文，恢复将要运行下一任务的上下文，并运行下一任务，这个过程称为上下文切换。在任务运行时，其上下文是高度动态的。调度器从一个任务切换到另一个任务所需要的时间称为上下文切换开销。

（7）可重入性。指一段代码被一个以上的任务调用，而不必担心数据的破坏。具有可重入性的函数任何时候都可以被中断，一段时间以后继续运行，相应数据不会遭到破坏。

（8）分发器。分发器是调度器的一部分，执行上下文切换并改变执行的流程。分发器完成上下文切换的实际工作并传递控制。任何时候，执行的流程通过三个区域之一：应用任务、中断服务程序（Interrupt Service Routines，ISR）或内核。

根据如何进入内核的情况，分发的情况也有所不同。当一个任务是用系统调用时，分发器通常在每个任务的系统调用完成后退出内核。在这种情况下，分发器通常以调用-调用为基础的，因此，它可以协调由此引起的任何系统调用的任务状态转移。另一方面，如果一个 ISR 做系统调用，则分发器将被越过，直到 ISR 全部完成它的执行。

（9）调度算法。调度算法是调度器决定何时运行哪个任务的策略。

（10）优先级。每个任务都有其优先级，任务越重要，赋予的优先级就越高。

2．调度算法分类

当前，大多数内核支持两种普遍的调度算法，即基于优先级的抢占调度算法和时间轮转调度算法。

（1）基于优先级的抢占调度（Preemptive Priority-Based Scheduling）。优先级一般又可以分为静态优先级和动态优先级。应用程序在执行的过程中诸任务的优先级固定不变，称之为静态优先级。在静态优先级系统中，各任务以及它们的时间约束在程序编译时是已知的；应用程序在执行的过程中诸任务的优先级可以动态改变，称之为动态优先级。

这种类型的调度，在任何时候运行的任务是所有就绪任务中具有最高优先级的任务，任务在创建时被赋予了优先级，任务的优先级可以由内核的系统调用动态更改，这使得嵌入式应用对于外部事件的响应更加灵活，从而建立真正的实时响应系统。

（2）时间轮转调度（Round-Robin Scheduling）。为每个任务提供确定份额的 CPU 执行时间。

3．任务优先级分配方法

因为实时系统相当复杂，所以，给任务确定优先级并非易事。实时系统大多综合了软实时和硬实时这两种需求，软实时系统只要求任务尽快执行，并不要求在某一特定时间内完成，硬实时系统中，要求任务不但要正确无误执行，而且还要准时完成。

一般，可以采用单调执行速率调度法（Rate Monotonic Scheduling，RMS）来给任务

分配优先级，其规则是：执行最频繁的任务优先级最高。RMS 做了如下假设：

（1）所有的任务都是周期性的。

（2）任务间不需要同步，没有共享资源，没有任务间的数据交换等问题。

（3）系统采用抢占式调度，总是优先级最高且就绪的任务被执行。

（4）任务的死线是其下一周期的开始。

（5）每个任务具有不随时间变化的定常时间。

（6）所有的任务具有同等重要的关键性级别。

（7）非周期性任务不具有硬死线。

要使一个具有 n 个任务的实时系统中的所有任务都满足硬实时条件，必须使下述 RMS 定理成立：

$$\sum_i \frac{E_i}{T_i} \leqslant n\left(2^{1/n}-1\right)$$

式中，E_i是任务 i 的最长执行时间，T_i是任务 i 的执行周期。也就是说，E_i/T_i 是任务 i 所需的 CPU 时间。基于 RMS 定理，要所有的任务满足硬实时条件，所有有时间要求的任务总的 CPU 利用时间（或利用率）应当小于 70%。通常，作为实时系统设计的一条原则，CPU 利用率应当在 60%～70%之间。

例如，表 9-1 是一个采样系统的相关参数。

表 9-1　任务的相关参数

周期性任务	执 行 时 间	周　　期
Task 1	20	100
Task 2	25	150
Task 3	50	300

根据 RMS 定理，处理器利用率的计算如下：

$$\frac{20}{100}+\frac{25}{150}+\frac{50}{300} \leqslant 3(2^{1/3}-1)，\text{即}53.34\% \leqslant 77.98\%$$

该问题的利用率是 53%，低于理论边界 78%，定理的条件满足，则这三个任务的系统是可调度的，即每个任务都满足时限的要求。

4．时间轮转调度

纯粹的时间轮转调度是不能满足实时系统的要求的。取而代之的是，基于优先级抢占式扩充时间轮转调度，对于优先级相同的任务使用时间片获得相等的 CPU 执行时间。内核在满足以下条件时，把 CPU 控制权转交给下一个就绪态的任务。

（1）当前任务已无事可做。

（2）当前任务的时间片还没用完任务就已经结束了。

9.3.4 内核对象

实时嵌入式操作系统的用户可以使用内核对象来解决实时系统设计中的问题，例如，并发、同步与互斥、数据通信等。内核对象包括信号量、消息队列、管道、事件与信号等。

1．信号量

为了同步一个应用的多个并发线程和协调它们对共享资源的互斥访问，内核提供了一个信号量对象和相关的信号量管理服务。

信号量是一个内核对象，就像一把锁，任务获取了该信号量就可以执行期望的操作或访问相关资源，从而达到同步或互斥的目的。

信号量可以分为如下三类：

（1）二值信号量。二值信号量只能有两个值：0 或 1。当其值为 0 时，认为信号量不可使用。当其值为 1 时，认为信号量是可使用的。当二值信号量被创建时，既可以初始化为可使用的，也可以初始化为不可使用的。二值信号量通常作为全局资源，被需要信号量的所有任务共享。

（2）计数信号量。计数信号量使用一个计数器赋予一个数值，表示信号量令牌的个数，允许多次获取和释放。初始化时，如果计数值为 0，表示信号量不可用，否则计数值大于 0，表示信号量可用。每获取一次信号量其计数值就减 1，每释放一次信号量其计数值就加 1。在有些系统中，计数信号量允许实现的计数是有界的，有些则无界。同二值信号量一样，计数信号量也可用做全局资源。

（3）互斥信号量。互斥信号量是一个特殊的二值信号量，它支持所有权、递归访问、任务删除安全和优先级反转，以避免互斥固有的问题。互斥信号量初始为开锁状态，被任务获取后转到闭锁状态，当任务释放该信号量时又返回开锁状态。

通常，内核支持创建和删除信号量操作、获取和释放信号量操作、清除信号量的等待队列操作以及获取信号量信息操作。

2．消息队列

多数情况下，任务活动同步并不足以满足实时响应的要求，任务之间还必须能够交换信息。为了实现任务之间的数据交换，内核提供了消息队列对象和消息队列的管理服务。

消息队列是一个类似于缓冲区的对象，通过它任务和 ISR 可以发送和接收消息，实现数据的通信。消息队列暂时保存来自发送者的消息，直到有接受者准备读取这些消息为止。

消息队列建立时，内核给消息队列分配唯一 ID 号，并为之创建队列控制块（Queue Control Block，QCB）和任务等待列表，并根据用户参数为消息队列分配内存。

消息的发送和接收过程是，在某些内核的实现中，当一个任务试图给一个满的消息

队列发送消息时，发送函数会给该任务返回一个错误代码，有些内核则允许这样的任务被阻塞，将该任务移入发送等待列表中。

大多数内核支持典型的消息队列操作，包括创建和删除消息队列、发送和接收消息以及获取消息队列的信息等。

3．管道

管道是提供非结构化数据交换和实现任务同步的内核对象。每个管道有两个端口，一端用来读，一端用来写。数据在管道中就像一个非结构的字节流，数据按照先进先出方式从管道中读出。

一般嵌入式操作系统内核支持两类管道对象：

（1）命名管道。具有一个类似于文件名的名字，像一个文件或设备出现在文件系统中，需要使用命名管道的任何任务或 ISR 都可以用误名字对其引用。

（2）匿名管道。一般动态创建，且必须使用创建时返回的描述符才可引用此类型的管道。

通常，管道支持创建和删除一个管道、读/写管道、管道控制、管道上的轮询等操作。

4．事件

某些特殊的嵌入式操作系统提供一个特殊的寄存器作为每个任务控制块的一部分，称为事件寄存器。它是一个属于任务的对象，并由一组跟踪指定事件的二值事件标志组成。

一般，嵌入式操作系统支持事件寄存器机制，创建一个任务时，内核同时创建一个事件寄存器作为任务控制块的一部分。经过事件寄存器，一个任务可以检查控制它执行的特殊事件是否出现。一个外部源，如另一个任务或中断处理程序，可以设置该事件寄存器的位，通知任务一个特殊事件的发生。任务说明它所希望接收的事件组，这组事件保存在寄存器中，同样，到达的事件也保存在接收的事件寄存器中。另外，任务还可以指示一个时限说明它愿意等待某个事件多长时间。如果时限超过，没有指定的事件达到任务，则内核唤醒该任务。

希赛教育专家提示：在进行应用设计的时候，事件寄存器的事件是不排队的，不能累计相同事件发生的次数。

5．信号

信号是当一个事件发生时产生的软中断，它将信号接收者从其正常的执行路径移开并触发相关的异步处理。本质上，信号通知其他任务或 ISR 运行期间发生的事件，与正常中断类似，这些事件与被通知的任务是异步的。信号的编号和类型依赖于具体的嵌入式的实现。

通常，嵌入式均提供信号设施，任务可以为每个希望处理的信号提供一个信号处理程序，或是使用内核提供的默认处理程序，也可以将一个信号处理程序用于多种类型的信号。信号可以有被忽略、挂起、处理或阻塞等 4 种不同的响应处理。

6．条件变量

条件变量是一个与共享资源相关的内核对象，它允许一个任务等待其他任务创建共享资源需要的条件。一个条件变量实现一个谓词，谓词是一组逻辑表达式，涉及共享资源的条件。谓词计算的结果是真或假，如果计算为真，则任务假定条件被满足，并且继续运行，反之，任务必须等待所需要的条件。当任务检查一个条件变量时，必须原子性地访问，所以，条件变量通常跟一个互斥信号量一起使用。

一个任务在计算谓词条件之前必须首先获取互斥信号量，然后计算谓词条件，如果为真，条件满足继续执行后续操作。否则，不满足，原子性地阻塞该任务并先释放互斥信号量。

条件变量不是共享资源同步访问的机制，大多数开发者使用它们让任务等待一个共享资源到达一个所需的状态。

9.3.5　内核服务

嵌入式操作系统的内核服务主要有异常和中断、计时器、I/O 管理等。

1．异常和中断

大多数嵌入式处理器体系结构都提供了异常和中断机制，允许处理器中断正常的执行路径。这个中断可能由应用软件触发，也可由一个错误或不可预知的外部事件来触发。而大多数嵌入式操作系统则提供异常和中断处理的包裹功能，使嵌入式系统开发者避免底层细节。

（1）异常。指任何打断处理器正常执行，迫使处理器进入特权执行模式的事件。

（2）中断。也称为外部中断，是一个由外部硬件产生的事件引起的异步异常，大多数嵌入式处理器体系结构中将中断归为异常的一类。

（3）同步异常。程序内部与指令执行相关的事件引起异常，如内存偶地址校准异常、除数为零异常。

（4）异步异常。与程序指令不相关的外部事件产生的异常，如系统复位异常、数据接收中断。

从应用的观点来看，异常和外部中断是外部硬件和应用程序通信的一种机制。一般来讲，异常和中断可以在如下两个方面应用于应用设计：内部错误和特殊条件管理、硬件并发和服务请求管理。

（1）内部错误和特殊条件管理。对大范围的错误进行处理和适当复原且不能导致停机，是嵌入式系统应用领域的重要课题。内部错误发生时，例如出现被零除、溢出或其他运算错误的情况下，计算任务停止运行，开始执行异常服务例程。

在特殊条件下，如陷入指令产生异常时，也会使处理器进入特权模式运行。在软件调试时，由处理器的断点特性而引起的追踪异常，可以让调试器代理在嵌入式系统上作特殊处理和单步跟踪等操作。

（2）硬件并发和服务请求管理。同时执行不同类型工作的能力对于嵌入式系统来说是十分重要的。许多外部硬件都可以与处理器并行地工作。因此，硬件并发和外部中断在嵌入式应用设计中用途广泛。如，利用外部中断可以给嵌入式处理器发信号或通知它外部硬件正在请求一个服务，当系统的网络接口硬件收到一个数据包时就可以用中断指出数据包已经到达。

目前，大多数嵌入式处理器都具有多个异常类型，按优先级从高到低的排列为：异步不可屏蔽、同步精确、同步不精确、异步可屏蔽。可以被软件阻塞或开放的异步异常称为可屏蔽的异常，否则，为不可屏蔽异常。不可屏蔽的异常总是被处理器处理，如硬件复位异常。许多处理器具有一个专门的不可屏蔽中断请求线，任何连接到该请求线的硬件都可以产生不可屏蔽中断。精确异常指处理器的程序计数器可以精确地指出引起异常的指令。而在流水线或指令预取的处理器上则不能精确地判断引起异常的指令或数据，这时的异常称为不精确异常。

从应用程序的观点看，所有的异常都具有比操作系统内核对象更高的优先级，包括任务、队列和信号量等。

实时内核最重要的指标是中断关了多长时间。所有的实时系统在进入临界代码时都要关中断，执行完临界代码之后再开中断。

中断延迟时间=关中断的最长时间+开始执行中断服务子程序的第一条指令的时间

中断恢复时间是微处理器返回到被中断的程序代码所需要的时间。

2．计时器

计时器是实时嵌入式系统的一个组成部分。时间轮转调度算法、存储器定时刷新、网络数据包的超时重传以及目标机监视系统的时序等都严格依赖于计时器。许多嵌入式系统用不同形式的计时器来驱动时间敏感的活动，即硬件计时器和软件计时器。硬件计时器是从物理计时芯片派生出来的，超时后可以直接中断处理器，硬件计时器对精确的延迟操作具有可预测的性能。而软件计时器是通过软件功能调度的软件事件，能够对非精确的软件事件进行有效的调度，通常具有以下特性：

- 高效的计时器维护，即计时器倒计数。
- 高效的计时器安装，即计时器启动。
- 高效的计时器删除，即停止一个计时器。

另外，使用软件计时器可以减轻系统的中断负担。

3．I/O 管理

通常，一个嵌入式系统是为处理与设备相关的需求而设计的。几乎所有的嵌入式系统都有一些 I/O 形式的操作，这些 I/O 操作运行在不同的设备上。

从系统开发者的观点看，I/O 操作意味着与设备的通信，对设备初始化、执行设备与系统之间的数据传输以及操作完成后通知请求者；从系统的观点看，I/O 操作意味着对请求定位正确的设备，对设备定位正确的驱动程序，并保证对设备的同步访问。

I/O 设备、相关的驱动程序等共同组合成嵌入式系统的 I/O 子系统。图 9-5 所示是一个典型的微内核系统的层次模型图。

I/O 子系统定义一组标准的 I/O 操作函数，以便于对应用隐藏设备的特性。所有的设备驱动程序都符合并支持这个函数集，给应用提供一个能够跨越各种类型 I/O 的设备的统一的接口。

I/O 子系统通常维护一个统一的设备驱动程序表，使用 I/O 子系统的工具函数，可以将任何驱动程序安装到此表或从表中删除。另外，还使用一个设备表来跟踪为每个设备所创建的实例。

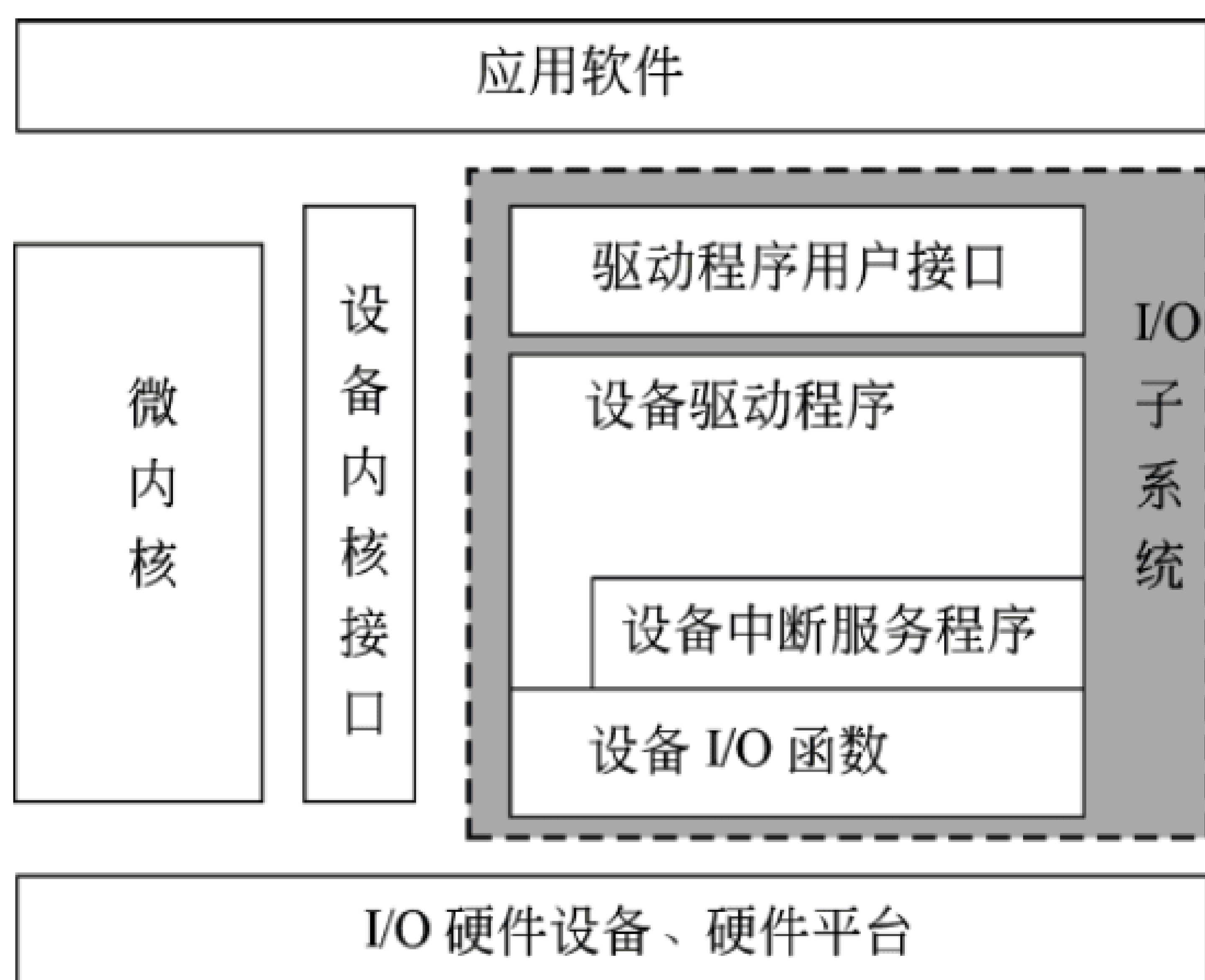


图 9-5 I/O 子系统层次模型

9.4 嵌入式系统分析与设计

嵌入式系统的分析与设计方法也遵循软件工程的一般原则，许多成熟的分析和设计方法都可以在嵌入式领域得到应用。嵌入式系统的开发过程同样也包括需求分析、系统设计、实现和测试几个基本阶段，但是每个阶段都有其独特的特征和重点。

与通常的系统设计相比，嵌入式系统设计具有以下特点：软、硬件协同并行开发；微处理器的类型多种多样；实时嵌入式操作系统的多样性；与台式机相比，可利用系统资源很少；应用支持少；要求特殊的开发工具；软、硬件都应很健壮；调试很困难。

嵌入式系统设计的常用指标如下。

（1）NRE 成本（非重发性工程成本）：设计系统所需要支付的一次性货币成本，即一旦设计完毕，不需要支付额外的设计费用，就可以制造任意数目的产品。

（2）单位成本：生产单个产品所需要支付的货币成本，不包含 NRE 成本。

（3）大小：指系统所占的空间，对软件而言，一般用字节数来衡量；对硬件而言则用逻辑门或晶体管的数目来衡量。

（4）性能：系统完成规定任务所需要的时间，是设计时最常用的设计指标，主要有两种衡量方式：一是响应时间，即开始执行到任务结束之间的时间。二是完成量，即单位时间内所完成的任务量。

（5）功率：系统所消耗的功率，它决定了电池的寿命或电路的散热需求。

（6）灵活性：在不增加 NRE 成本前提下，改变系统功能的能力。

（7）样机建立时间：建立系统可运行版本所需的时间，系统样机可能比最终产品更大更昂贵，但可以验证系统的用途和正确性，并可以改进系统的功能。

（8）上市时间：从系统开发到可以上市卖给消费者的时间，最主要的影响因素包括

设计时间、制造时间和检测时间。

（9）可维护性：系统推出或上市后进行修改的难易程度，特别是针对非原始开发人员进行修改。

（10）正确性：正确实现了系统的功能，可以在整个设计过程中检查系统的功能，也可以插入测试电路检验是否正确。

（11）安全性：系统不会造成伤害的概率。

各个设计指标之间一般是互相竞争的，改良了某个指标常常会导致其他指标的恶化，为了最好的满足设计最佳化，设计师必须了解各种软、硬件实现技术，并且能够从一种技术转移到另一种技术，以便找到特定约束下的最佳方案。

9.4.1　核心技术

嵌入式系统的核心技术主要有三种，分别是处理器技术、IC 技术、设计/验证技术。

1．处理器技术

处理器技术与实现系统功能的计算引擎结构有关，很多不可编程的数字系统也可以视为处理器，这些处理器的差别在于其面向特定功能的专用化程度，导致其设计指标与其他处理器不同。

2．IC 技术

（1）全定制。在全定制 IC 技术中，需要根据特定的嵌入式系统的数字实现来优化各层设计人员从晶体管的版图尺寸、位置、连线开始设计以达到芯片面积利用率高、速度快、功耗低的最优化性能。利用掩膜在制造厂生产实际芯片，全定制的 IC 设计也常称为大规模集成电路设计，具有很高的 NRE 成本、很长的制造时间，适用于大量或对性能要求严格的应用。

（2）半定制。半定制是一种约束型设计方法，包括门阵列设计法和标准单元设计法。它是在芯片制作好一些具有通用性的单元元件和元件组的半成品硬件，设计师仅需要考虑电路的逻辑功能和各功能模块之间的合理连接即可。这种设计方法灵活方便、性价比高，缩短了设计周期，提高了成品率。

（3）可编程。可编程器件中所有各层都已经存在，设计完成后，在实验室里即可烧制出设计的芯片，不需要 IC 厂家参与，开发周期显著缩短。可编程 ASIC 具有较低的 NRE 成本，单位成本较高，功耗较大，速度较慢。

3．设计/验证技术

嵌入式系统的设计技术主要包括硬件设计技术和软件设计技术两大类。其中，硬件设计领域的技术主要包括芯片级设计技术和电路板级设计技术两个方面。

芯片级设计技术的核心是编译/综合、库、测试/验证。编译/综合技术使设计师用抽象的方式描述所需的功能，并自动分析和插入实现细节。库技术将预先设计好的低抽象级实现用于高级。测试/验证技术确保每级功能正确，减少各级之间反复设计的成本。

软件设计技术的核心是软件语言。软件语言经历了从低级语言（机器语言、汇编语言）到高级语言（如结构化设计语言、面向对象设计语言）的发展历程，推动其发展的是汇编技术、分析技术、编译/解释技术等诸多相关技术。软件语言的级别也从实现级、设计级、功能级逐渐向需求级语言发展过渡。

早期，随着通用处理器概念的逐渐形成，软件技术迅速发展，软件的复杂度也开始增加，软件设计和硬件设计的技术和领域完全分开。设计技术和工具在这两个领域同步得到发展，也使得行为描述可以在越来越抽象的级别上进行，以适应设计复杂度不断增长的需要。这种同步发展如今又使得这两个领域都使用同样的时序模型来描述行为，因此这两个领域再度统一为一个领域即将成为可能。

9.4.2 设计流程

设计流程是系统设计期间应遵循的一系列步骤，其中一些步骤可以由自动化工具完成，而另外一些只可用手工完成。在嵌入式系统领域，有如下几种常用开发过程模型。

（1）瀑布模型。瀑布模型主要由 5 个主要阶段构成：需求分析阶段确定目标系统的基本特点；系统结构设计阶段将系统的功能分解为主要的构件；编码阶段主要进行程序的编写和调试；测试阶段检测错误；维护阶段修改代码以适应环境的变化，并改正错误和升级。各个阶段的工作和信息总是由高级的抽象到较详细的设计步骤单向流动，是一个理想的自顶向下的设计模型。

（2）螺旋模型。螺旋模型假定要建立系统的多个版本，早期的版本是一个简单的试验模型，用于帮助设计师建立对系统的直觉和积累开发此系统的经验，随着设计的进展，会创建更加复杂的系统。在每一层设计中，设计师都会经过需求、结构设计、测试三个阶段。在后期，当构成更复杂的系统版本时，每一个阶段都会有更多的工作，并需要扩大设计的螺旋，这种逐步求精的方法使设计师可以通过一系列的设计循环加深对所开发的系统的理解。螺旋的顶部第一个循环是很小很短的，而螺旋底部的最后的循环加入了对螺旋模型的早期循环的细节补充，螺旋模型比瀑布模型更加符合实际。

（3）逐步求精模型。逐步求精的模型是一个系统被建立多次，第一个系统被作为原型，其后逐个系统将进一步被求精。当设计师对正在建造的系统的应用领域不是很熟悉时，这个方法很有意义。通过建造几个越加复杂的系统，从而精炼系统使设计师能检验体系结构和设计技术。此外，各种迭代技术也可仅被局部完成，直到系统最终完成。

（4）层次模型。许多嵌入式系统本身是由更多的小设计组成的，完整的系统可能需要各种软件构件、硬件构件。这些部件可能由尚需设计的更小部件组成，因此从最初的完整系统设计到为个别部件的设计，设计的流程随着系统的抽象层次的变化而变化，从最高抽象层次的整体设计到中间抽象层次的详细设计，再到每个具体模块的设计，都是逐层展开的，其中每个流程可能由单个设计人员或设计小组来承担，每个小组依靠其他小组的结果，各个小组从上级小组获得要求，同时上级小组依赖于各个分组设计的质量

和性能。而且，流程的每个实现阶段都是一个从规格说明到测试的完整流程。

嵌入式系统设计不同于传统的软件设计，经常包含硬件设计和软件设计，图 9-6 是一个常用的嵌入式系统设计方法，其中前端活动，如规格说明和系统架构，需要同时考虑硬件和软件两个方面。类似地，后端设计如系统集成和测试要考虑整个系统。中间阶段，软件和硬件构件的开发彼此相互独立，并且大多数的硬件和软件的工作能够相对独立地进行。

9.4.3　硬件子系统设计

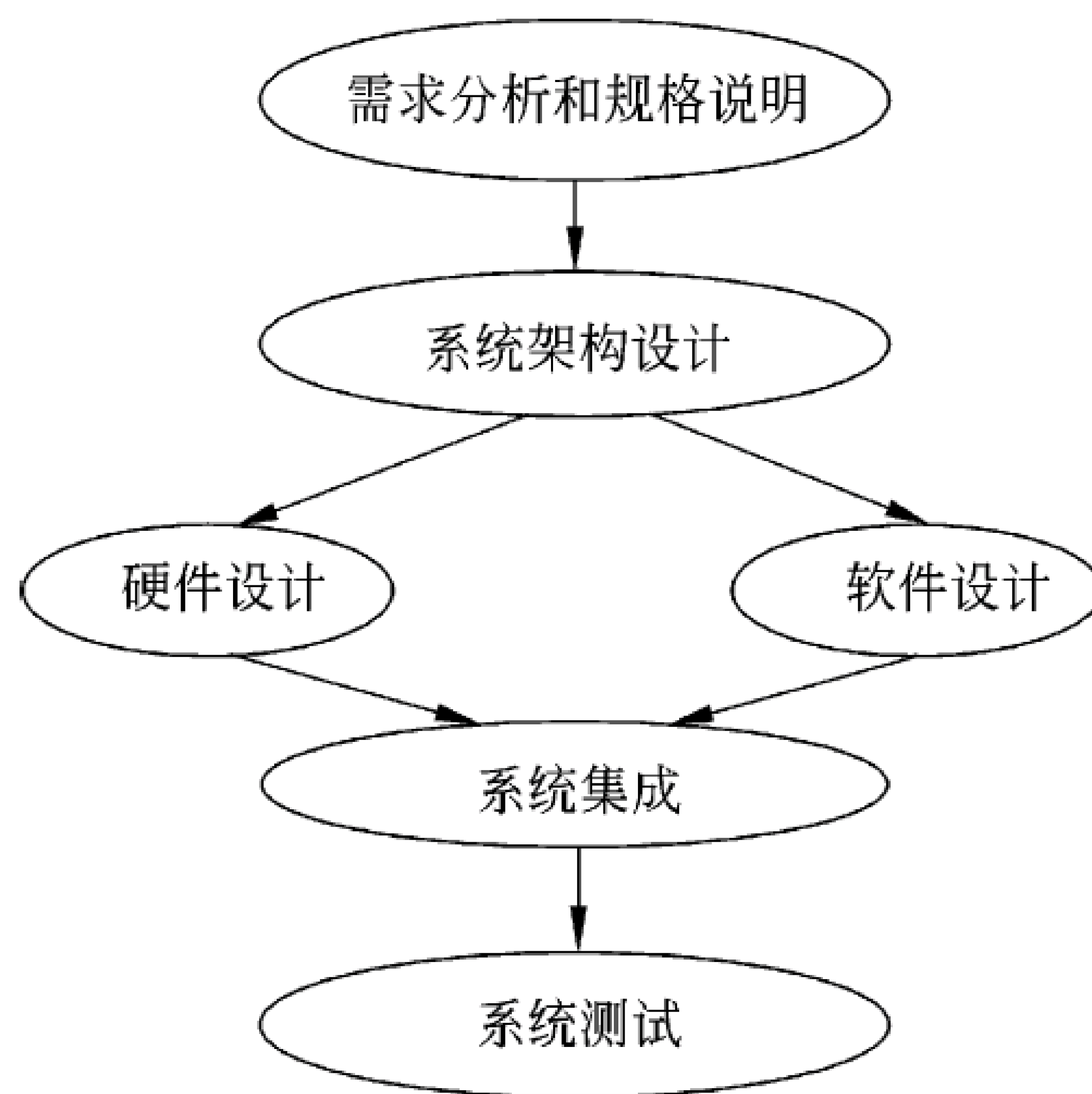


图 9-6　嵌入式系统常用的设计方法

嵌入式系统的开发环境由 4 部分组成：目标硬件平台、嵌入式操作系统、编程语言和开发工具，其中处理器和操作系统的选择应当考虑更多的因素，避免错误的决策影响项目的进度。

1．选择处理器技术

嵌入式系统设计的主要挑战是如何使互相竞争的设计指标同时达到最佳化。设计师必须对各种处理器技术和 IC 技术的优缺点加以取舍。一般而言，处理器技术与 IC 技术无关，也就是说任何处理器技术都可以使用任何 IC 技术来实现，但是最终器件的性能、NRE 成本、功耗、大小等指标会有很大的差异。更通用的可编程技术提供了较大的灵活性，降低了 NRE 成本，建立产品样机与上市的时间较快。定制的技术能够提供较低的功耗、较好的性能、更小的体积和大批量生产时的低成本。

通常，一个公司要推出一种产品，如机顶盒、家庭路由器或通用处理器等，可以先推出半定制产品，以尽快占领市场，然后再推出全定制的产品。也可先用较可靠的老技术实现处理器，再用新的技术实现下一代。同样，嵌入式系统的设计师可以使用可编程的器件来建立样机，以加速上市时间，批量时再采用定制器件。

根据这些原则，设计师便可以对采用的处理器技术和处理器做出合理选择。一般，全定制商品化的“通用处理器+软件”是大多数情况下都适用的一个选择。

2．通用嵌入式处理器的选择

根据用户的需求和项目的需要选择合适的通用嵌入式处理器，选择时需要考虑如下指标和标准。

（1）处理器的处理速度。一个处理器的性能取决于多个方面的因素：时钟频率，内部寄存器的大小，指令是否对等处理所有的寄存器等。对于许多需用处理器的嵌入式系统设计来说，目标不是在于挑选速度最快的处理器，而是在于选取能够完成作业的处理

器和 I/O 子系统。处理器的性能满足系统的需求，并有一定的余量，但也不必选得太高。

（2）技术指标。当前，许多嵌入式处理器都集成了外围设备的功能，从而减少了芯片的数量，进而降低了整个系统的开发费用。开发人员首先考虑的是，系统所要求的一些硬件能否无需过多的组合逻辑（GL）就可以连接到处理器上。其次是考虑该处理器的一些支持芯片，如 DMA 控制器、内存管理器、中断控制器、串行设备、时钟等的配套。

（3）开发人员对处理器的熟悉程度，即项目的开发人员需要在处理器本身的成本和开发成本之间做一个权衡。

（4）处理器的 I/O 功能是否满足系统的需求，即许多处理器提供内置的外部设备，以减少芯片数量、降低成本，应尽量考虑这种方案。

（5）处理器的相关软件支持工具，即该款处理器是否具有完善的嵌入式操作系统、编程语言和开发工具的支持等。

（6）处理器的调试，即处理器是否集成了调试功能，如是否支持 JTAG、BDM 等调试方式。

（7）处理器制造商的支持可信度。在产品的生命周期，在选择某种处理器时，设计师必须确认它有足够的供货量、技术支持等，是否是低功耗的产品。

嵌入式微处理器最大并且增长最快的市场是手持设备、电子记事本、PDA、手机、GPS 导航器、智能家电等消费类电子产品，这些产品中选购的微处理器典型的特点是要求高性能、低功耗。许多 CPU 生产厂家已经进入了这个领域。

3．硬件设计的注意事项

首先，将硬件划分为部件或模块，并绘制部件或模块连接框图。其次，对每个模块进行细化，把系统分成更多个可管理的小块，可以被单独实现。通常，系统的某些功能既可用软件实现也可用硬件实现，没有一个统一的方法指导设计师决定功能的软硬件分配，但是可以根据约束清单，在性能和成本之间进行权衡。

设计软、硬件之间的接口，接口的设计需要硬件设计师和软件设计师协同工作才能完成，良好的接口设计可以保证硬件简洁、易于编程。设计时需要注意以下几点。

（1）I/O 端口：列出硬件的所有端口、端口地址、端口属性、使用的命令和序列的意义、端口的状态及意义。

（2）硬件寄存器：对每个寄存器设计寄存器的地址、寄存器的位地址和每个位表示意义以及对寄存器读写的说明、使用该寄存器的要求和时序说明。

（3）内存映射：共享内存和内存映射 I/O 的地址，对每个内存映射，说明每个 I/O 操作的读/写序列、地址分配。

（4）硬件中断：如何使用硬件中断，列出所使用的硬件中断号和分配的硬件事件。

（5）存储器空间分配：列出系统中程序和数据占用的空间大小、位置，以及存储器类型和访问方式等。

总之，硬件设计师应该给软件设计师更多、更详细的信息，便于进行软件设计和

开发。

9.4.4　软件子系统设计

根据需求分析阶段的规格说明文档，确定系统计算模型，对软件部分进行合理的设计即可，主要内容将在后面章节中介绍。

1．嵌入式操作系统的选择

在选择嵌入式操作系统时，也需要做多方面的考虑：

（1）操作系统的功能。根据项目需要的操作系统功能来选择操作系统产品，要考虑系统是否支持操作系统的全部功能还是部分功能，是否支持文件系统、人机界面，是实时系统还是分时系统以及系统是否可裁减等因素。

（2）配套开发工具的选择。有些实时操作系统只支持该系统供应商的开发工具。也就是说，还必须向操作系统供应商获取编译器、调试器等。有些操作系统使用广泛且有第三方工具可用，因此，选择的余地比较大。

（3）操作系统的移植难易程度。操作系统到硬件的移植是一个重要的问题。它是关系到整个系统能否按期完工的一个关键因素，因此要选择那些可移植性程度高的操作系统，从而避免操作系统难以向硬件移植而带来的种种困难，加速系统的开发进度。

（4）操作系统的内存需求如何。均衡考虑是否需要额外 RAM 或 EEPROM 来迎合操作系统对内存的较大要求。有些操作系统对内存的要求是目标相关的。如 Tornado/VxWorkx，开发人员能按照应用需求分配所需的资源，而不是为操作系统分配资源。从需要几 K 字节存储区的嵌入设计到需求更多的操作系统功能的复杂的高端实时应用，开发人员可任意选择多达 80 种不同的配置。

（5）操作系统附加软件包。是否包含所需的软件部件，如网络协议栈、文件系统、各种常用外设的驱动等。

（6）操作系统的实时性。实时性分为软实时和硬实时。有些嵌入式操作系统只能提供软实时，如 Microsoft Windows CE 2.0 是 32 位，Windows 兼容，小内核，可伸缩实时操作系统，满足大部分嵌入式和非嵌入式应用的需要。但不够实时，属于软实时嵌入式操作系统。

（7）操作系统的灵活性如何。操作系统是否具有可剪裁性，即能否根据实际需要进行系统功能的剪裁。有些操作系统具有较强的可剪裁性，如嵌入式 Linux、Tornado/VxWorks 等。

2．编程语言的选择

在编程语言的选择上，要注意以下问题：

（1）通用性。随着微处理器技术的不断发展，其功能越来越专用，种类越来越多，但不同种类的微处理器都有自己专用的汇编语言。这就为系统开发者设置了一个巨大的障碍，使得系统编程更加困难，软件重用无法实现，而高级语言一般和具体机器的硬件

结构联系较少，比较流行的高级语言对多数微处理器都有良好的支持，通用性较好。

（2）可移植性。由于汇编语言和具体的微处理器密切相关，为某个微处理器设计的程序不能直接移植到另一个不同种类的微处理器上使用，因此，移植性差。高级语言对所有微处理器都是通用的，因此，程序可以在不同的微处理器上运行，可移植性较好。这是实现软件重用的基础。

（3）执行效率。一般来说，越是高级的语言，其编译器和开销就越大，应用程序也就越大、越慢。但单纯依靠低级语言，如汇编语言来进行应用程序的开发，带来的问题是编程复杂、开发周期长。因此存在一个开发时间和运行性能间的权衡。

（4）可维护性。低级语言如汇编语言，可维护性不高。高级语言程序往往是模块化设计，各个模块之间的接口是固定的。因此，当系统出现问题时，可以很快地将问题定位到某个模块内，并尽快得到解决。另外，模块化设计也便于系统功能的扩充和升级。

（5）基本性能。在嵌入式系统开发过程中使用的语言种类很多，比较广泛应用的高级语言有 Ada、C/C++、Modula-2 和 JAVA 等。Ada 语言定义严格，易读易懂，有较丰富的库程序支持，目前在国防、航空、航天等相关领域应用比较广泛，未来仍将在这些领域占有重要地位。C 语言具有广泛的库程序支持，目前在嵌入式系统中是应用最广泛的编程语言，在将来很长一段时间内仍将在嵌入式系统应用领域占重要地位。C++是一种面向对象的编程语言，目前在嵌入式系统设计中也得到了广泛的应用，如 GNU C++。Visual C++是一种集成开发环境，支持可视化编程，广泛应用于 GUI 程序开发。但 C 与 C++相比，C++的目标代码往往比较庞大和复杂，在嵌入式系统应用中应充分考虑这一因素。

3．软件开发过程

嵌入式软件的开发过程不同一般通用软件的开发过程，主要有如下步骤：

（1）选择开发语言，建立交叉开发环境。

（2）根据详细设计说明编写源代码，进行交叉编译、链接。

（3）目标代码的重定位和下载。

（4）在宿主机或目标机调试、验证软件功能。

（5）进行代码的优化。

在嵌入式产品的开发设计过程中，开发阶段完成系统产品的实现，这一阶段同时需要完成一系列的文档，这些文档对完成产品设计、维护相当重要，这些文档分别为技术文件目录、技术任务书、技术方案报告、产品规格、技术条件、设计说明书、试验报告、总结报告等。

4．系统集成与测试阶段

通常嵌入式系统测试主要包括软件测试、硬件测试、单元测试三个部分。一般系统的硬件测试包括可靠性测试和电磁兼容性测试，关于电磁兼容性目前已经有了强制性国内和国际标准。

嵌入式系统软件测试方法和原理跟通用软件的测试基本一致，软件测试时，一般需要测试实例或测试序列，序列有两种来源，一种是需要用户进行设计，另一种是标准的测试系列。无论哪种测试实例，都要求实例能够高概率发现最多的错误，但在测试的内容上有些差别。

（1）嵌入式软件必须长时间稳定运行。

（2）嵌入式软件一般不会频繁地版本升级。

（3）嵌入式软件通常使用在关键性的应用中。

（4）嵌入式软件必须和嵌入式硬件一起对产品的故障和可靠性负责。

（5）现实世界的条件是异步和不可预测的，使得模拟测试非常困难。

由于这些差别，使得嵌入式系统软件测试主要集中在如下 4 个不同的方面：

（1）因为实时性和同时性很难同时满足，所以大多数测试集中于实时测试。

（2）大多数实时系统都有资源约束，因此需要更多的性能和可用性测试。

（3）可以使用专用实时跟踪工具对代码覆盖率进行测试。

（4）对可靠性的测试级别比通用软件要高得多。

另外，性能测试也是设计嵌入式系统中需要完成的最主要的测试活动之一，对嵌入式系统有决定性的影响。

9.5　多任务设计的相关问题

当开发多任务的嵌入式系统应用时，许多普遍性的设计问题随之产生。因为系统的资源有限，多个任务执行时，共享和竞争相同的资源是不可避免的，在可抢占的多任务环境中，资源共享是任务优先级的一个函数，任务的优先级越高任务越重要。当访问共享资源时，高优先级的任务先于较低优先级任务，因此资源共享必须遵循这个规则，另外，资源的利用率最大化问题也是一个突出的问题。嵌入式系统中多个任务利用并发执行达到效率最大化，任务之间协同工作也是实际应用的普遍要求，因此任务间的通信和同步问题也是设计师必须考虑的。

9.5.1　标识设备的依赖性

嵌入式系统的 I/O 设备可以分为主动设备与被动设备两大类：

（1）主动 I/O 设备（Active I/O Device）可以产生周期性的中断或与其他设备同步的中断，也可以产生非周期性的中断或与其他设备异步的中断。

（2）被动 I/O 设备（Passive I/O Device）不产生中断，必须由应用程序初始化与这些设备的通信。

主动设备产生中断表示设备已经准备就绪可以发送/接收数据或输入/输出过程已经处理完毕，给 CPU 发中断，请求进一步的处理。被动设备的输入/输出必须由应用程序

产生，才能跟设备交互，应用还可以周期性或非周期性地查询这些设备。当这些设备负责尖峰信号采样的处理时，应用程序的轮询频率要特别注意，若轮询频率太低，则尖峰和低谷可能被漏掉，若频率太高，则会使系统的负荷过重而浪费 CPU 周期。

1．标识主动设备

主动设备利用中断与实时应用进行通信，每当主动设备需要给实时应用发送或通知一个事件时，设备就产生一个中断，触发一个中断服务程序执行必要的处理，若该事件还需要进一步的处理，则中断服务例程通过通信机制，将该处理移交给一个专门任务负责。

主动 I/O 设备任务设计的建议如下：

（1）为单独的主动异步 I/O 设备指派单独的任务。

（2）对不经常产生中断且具有较长时限的 I/O 设备进行任务组合。

（3）为具有不同输入和输出速率的设备指派单独的任务。

（4）给中断产生设备相关的任务指派更高的优先权。

（5）指派一个资源控制任务控制对 I/O 设备的访问。

（6）为需要提交给多个任务的 I/O 请求指派一个事件分发任务。

2．标识被动设备

被动设备不同于主动设备，因为被动设备不产生中断，这些设备将一直处于空闲状态直到有任务请求到来。一个任务必须初始化事件或数据传输，任务通过轮询与这些设备通信，输入与输出的具体工作也要由任务来执行。

被动 I/O 设备任务设计的建议如下：

（1）当与这些被动设备的通信是非周期性的并且时限不太紧急时，给这些被动设备指派一个单独的任务。

（2）通过计时器事件触发对设备的轮询。

（3）对于轮询周期较短的任务应该分配相对较高的优先级。

（4）对每个需要不同速率轮询的被动设备指派一个单独的任务。

3．标识事件的依赖性

实时应用中的事件可能跨越多个任务进行传播，无论一个事件是从外部 I/O 设备还是从内部应用产生，都需要一个任务或一组任务来适当地处理这些事件。

4．标识时间依赖性

在设计实时应用之前，要弄清应用要求的每个时限，在标识时间时限完毕后，可以指派单独的任务处理每个单独的时限，依据每个时限的关键性和紧急性分配不同优先级。

（1）标识关键和紧急的活动。关键任务（Critical Task）是指其失败将引起灾难的任务，时限或长或短，但必须总要满足，否则系统将无法实现其要求。紧急任务（Urgent Task）是指一个时限相对较短的任务，其失败不一定引起灾难性后果。关键任务和紧急任务一般都要分配相对较高的优先级。但要注意关键性和紧急性的差别。

（2）标识不同周期的执行速率。每个速率驱动（RateDrive）的活动不依赖于任何其他速率而运行，可以标识周期性活动，并且可以用相同的速率将多个活动组合成多个任务。

（3）标识临时集合。虽然实时系统在功能上是不相关的，但可能包含总是同时执行的代码序列，这样的序列就表现出了临时聚合（Temporal Cohesion），如被相同的外部激励（如一个计时器）驱动的多个不同的活动，将这些活动序列组合成一个任务可以减轻系统的负载。

5．标识计算边界活动

在实时系统中，某些活动可能比其他活动需要更多的 CPU 时间，这些活动被称为计算性边界活动（Computationally Bound Activities），一般可能是数值处理活动，并且典型地具有相对较长的时限，这类活动通常分配较低的优先级，也可以按一般的优先级分配时间，当不需要运行更关键的任务时，可以处理这些任务。

6．标识功能聚合

功能聚合（Functional Cohesion）要求将执行相对紧密的几组活动函数或代码序列组合成一个单独的任务。另外，若两个任务紧密耦合（互相传递大量的数据），也应当考虑将它们组合成一个任务，这样将有助于消除同步问题和通信过载。

7．标识服务特定目的的任务

可以根据服务的特殊目的对任务进行分组，如一个安全任务，它探测可能的问题、设置警报，并且向用户发送通告，建立和执行正确的监控，这些都可以在一个安全任务中协调处理。

8．标识顺序聚合

顺序聚合（Sequential Cohesion）是将那些必须按照给定顺序发生的活动组合成一个任务，从而进一步强调操作顺序的重要性。一个典型的实例就是必须按照预定的次序执行计算的序列，某一步的输出将作为下一步的输入，以此类推。

9.5.2　资源请求模型

在嵌入式系统中，被各个并发执行的任务所共享资源，包括 I/O 设备、寄存器和内存区等，大致可分为如下两类。

（1）可抢占的（Preemptible）：这类资源暂时或偶然地被从任务中移走，不影响任务的执行状态和最终结果。多个任务共享处理器的寄存器就属于这种情况。

（2）不可抢占的（non-preemptible）：这类资源必须由占用它的任务放弃，否则将发生无法预测的后果。共享内存区域属于这一类型，读写任务在完成操作之前，不允许其他任务写入该内存区域。

在嵌入式实时操作系统中，死锁的形式取决于任务如何请求资源，即资源请求模型。当任务申请资源时，按照请求方式可以划分如下请求模型。

（1）单资源请求模型：指一个任务在任意给定时间内最多只能有一个明确的资源需求。

（2）与资源请求模型：指一个任务在任意给定时间内可以同时明确地提出多个资源请求。

（3）或资源请求：指一个任务可以请求一组资源，但只要这组资源中一个资源可以使用，任务就可以继续执行。

（4）与-或资源请求模型：指一个任务可以用 AND 和 OR 模型任何组合方式进行资源的请求。

9.5.3 死锁

死锁（deadlock）是指系统中执行的多个并发任务因资源的需求不能满足而产生的永久阻塞现象。

一个典型的实时系统中含有多种类型的资源和竞争这些资源的并发任务，每个任务可以在其生命周期内获得数目和类型不定的各种资源，死锁潜在地存在于系统之中，当满足如下 4 个基本必要条件时，可能发生死锁：互斥访问、不可抢占、保持并等待、循环等待。

实时系统中的死锁又可根据其特性分为稳态死锁和暂态死锁两种类型。

（1）稳态死锁（Stable Deadlock）是永久性的并且只有外部干预才能解除死锁状态。

（2）暂态死锁（Temporal Deadlock）是暂时死锁，根据时间约束，死锁集合中的一个或多个任务超过时限，则通过释放可能引起死锁的资源来消除死锁。

1．死锁的检测

死锁监测是实时操作系统周期性的监测算法，该算法检查当前资源的分配状态和未解决的请求，判断系统中是否有死锁。显然，一个系统若具有死锁检测能力，则资源的利用率更高。

通常设计死锁检测算法时需要考虑如下因素：

（1）对于 Single 请求模型中的死锁，资源图中的一个循环是充要条件。

（2）对于 AND 请求模型中的死锁，资源图中的一个循环是充要条件，并且一个任务包含在多个死锁集合中是可能的。

（3）对于 OR 请求模型中的死锁，资源图中的一个结是充要条件。（在资源图中，对于结点 A，如果 A 可到达的集合是所有结点 B 的集合，则存在一个从 A 到达 B 的有向路径，一个结点表示请求集合 K，而它的每个结点可精确到达的集合也是 K）。

（4）对于 AND-OR 模型中的死锁，通常，检查 OR 模型是否有结，然后，再检查 AND 模型是否存在死锁的循环。

2．死锁的恢复

当死锁检测出来后，下一步是恢复并找出消除死锁的方法，在解除死锁之前，必须

执行必要恢复算法。

对于可抢占的资源，资源抢占是一种恢复死锁的方法。当死锁检测算法构造死锁集合后，该集合传递给恢复算法，然后通过从任务中抢走资源并将这些资源重新分给其他任务，以恢复死锁。这个过程可以暂时打破死锁。对于可抢占资源的抢占不能直接影响任务的执行状态或最终结果，但是资源抢占可以影响任务的时间约束。资源抢占期间可能引起被强占任务的夭折，将导致一个未完成的执行并间接地影响任务的结束。

对于不可抢占资源，资源抢占对于被抢占的任务可能是有害的，并且可能影响其他任务的结果。如果任务有一个内在的、自我恢复机制，则不可抢占资源的抢占效果可以最小化。通过在执行路径上定义检查点，可以实现任务的自我恢复，一旦任务到达检查点，任务改变全局状态反映此迁移，另外，任务必须定义在任务允许恢复执行后能被死锁恢复算法调用的入口点。

3．死锁避免与预防

死锁避免是在资源分配时系统使用一个算法，保证当前的分配不会最终导致死锁。为了死锁避免能够工作，每个任务必须估计它对每个资源类型的最大需求量，这在动态系统中很难做到，然而，对于相对静态的嵌入式系统可以实现死锁避免。但是，由于这个估计是个高额估计，会导致高负载的系统较低的资源利用率。

死锁预防是在系统中构造一组约束和要求，以打破死锁的 4 个条件中一个或多个。死锁预防的方法如下：

（1）消除保持与等待死锁条件。任务在一个时间内请求所有需要的资源，仅当请求集合中的每个资源都得到满足任务才可以开始执行。这个要求避免了执行期间的等待，但这种方法在实现中受到了几个限制。首先，在动态的系统中，任务很难预测未来需要哪些资源，即使可以精确地预测，也无法保证这个预测集合中的资源都能使用到。另外一个隐含的缺点是所有的资源必须同时释放，资源必须被保持到任务执行结束，导致系统极低的资源利用率。

（2）消除不可抢占的死锁条件。如果一个任务占有一些资源，进而申请新的资源，但遭到拒绝，则该任务必须释放已经占有的资源，任务必须初始化一个新的请求，包括新的资源和原先已经占有的资源。但是，对于不可抢占资源，这种要求意味着每个任务必须从头开始或者从定义好的检查点重新执行，这个过程使得已经部分完成的工作无效，一个任务潜在地可能总也完成不了，这取决于系统在给定时间内已有任务的平均个数，取决于整个系统的调度行为。

（3）消除循环等待的死锁条件。必须对资源强加一个次序，使得如果一个任务占有资源 R_i，后续的请求必须是对资源 R_j 的请求，下一个请求必须是对资源 R_k 的请求，这里，$k>j>i$，以此类推。

9.5.4 优先级反转问题

优先级反转指由于资源竞争，低优先级的任务在执行，而高优先级的任务在等待的现象。当具有不同优先级的任务中存在相互依赖关系时，就可能发生优先级反转。

当系统内低优先级的任务 C 占用着高优先级任务 A 要使用的资源时，已经就绪高优先级的任务 A 只好等待低优先级的任务 C 执行完毕并释放该资源后才能被调度执行，这时，如果有中优先级任务 B 也进入就绪，剥夺了尚未完成的低优先级任务 C 的 CPU 使用权，使得系统只有先让 B 运行完毕且低优先级任务 C 重新运行结束并释放资源后，高优先级的任务 A 才能运行。任务 A 和 B 优先级发生了颠倒，在这种情况下，高优先级的任务的优先级实际上已经降到了低优先级的水平，从而发生优先级反转现象，这种情况下中优先级的任务抢占低优先级的任务，时间可能不确定，这时发生的优先级反转是无界优先级反转，如图 9-7 所示。

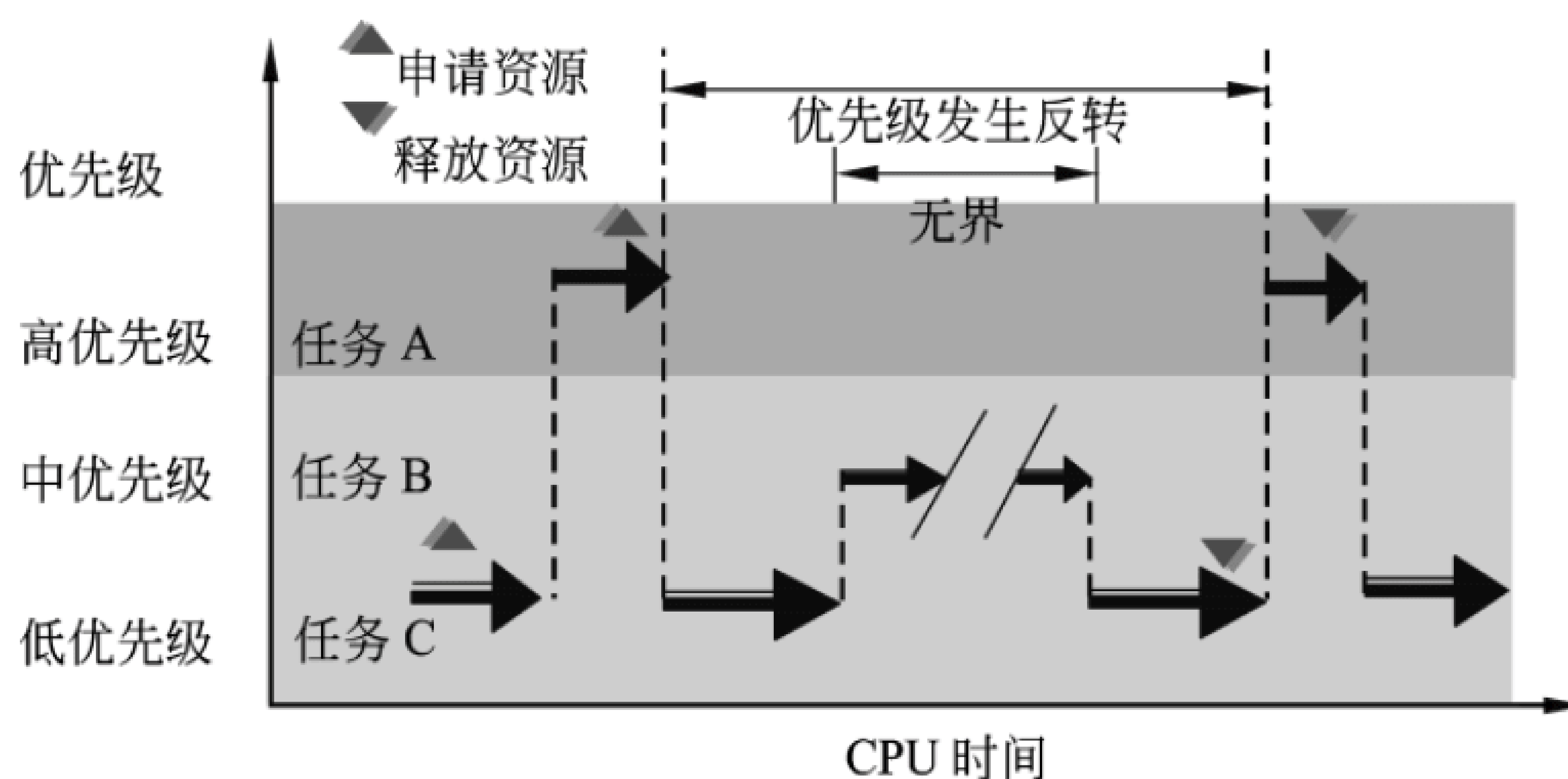


图 9-7 无界优先级反转实例

当一个较高优先级的任务请求一个较低优先级任务占有的资源时，较低优先级的任务却锁住了该资源，即使较高优先级的任务就绪，它也必须等待等优先级的任务释放资源后才能继续运行，这种情况下，低优先级的任务占用资源的时间是已知的，这时发生优先级反转是有界优先级反转，如图 9-8 所示。

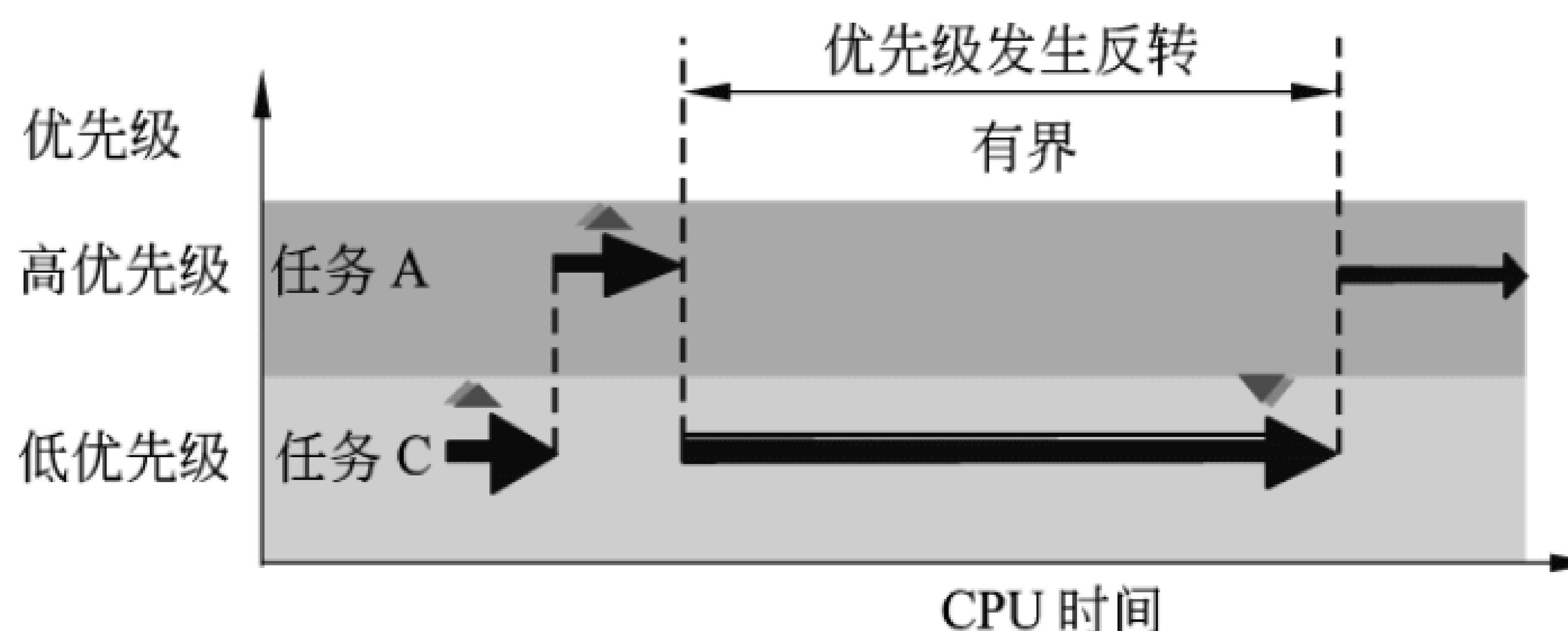


图 9-8 有界优先级反转实例

当优先级发生反转时，某些任务的执行时间减少，同时其他任务的执行时间延长，导致任务错失时限，进而引起时序反常。优先级反转是由不同优先级任务间的资源同步引起的，优先级的翻转不可避免，但可以使用资源控制协议将其降到最低限度。

为防止发生优先级的反转，多任务内核应允许动态地改变任务的优先级，如果一个任务占有着正要被高优先级任务请求的资源，那么该任务的优先级会暂时提升到与较高优先级任务相同的优先级水平，这叫做优先级继承。当然，改变任务优先级的时间开销是相当可观的。

1．优先级继承协议

优先级继承协议规则如表 9-2 所示。

表 9-2　优先级继承协议规则

协议规则号	描　　述
1	如果资源 R 在使用，则任务 T 被阻塞
2	如果资源 R 是空闲的，则资源 R 被分配给任务 T
3	当较高优先级的任务 T'请求相同的资源 R 时，任务 T 执行优先级被提升到请求任务 T'的优先级等级
4	当释放资源 R 后，任务 T 返回到先前的优先级

优先级继承协议是动态的，一个不相关的较高优先级任务仍可抢占任务，这是基于优先级、可抢占调度模式的本性的，并且任务优先级在反转期间，被提升优先级的任务的优先级可以继续被提升。但是，优先级继承协议不能消除死锁。优先级继承的例子如图 9-9 所示。

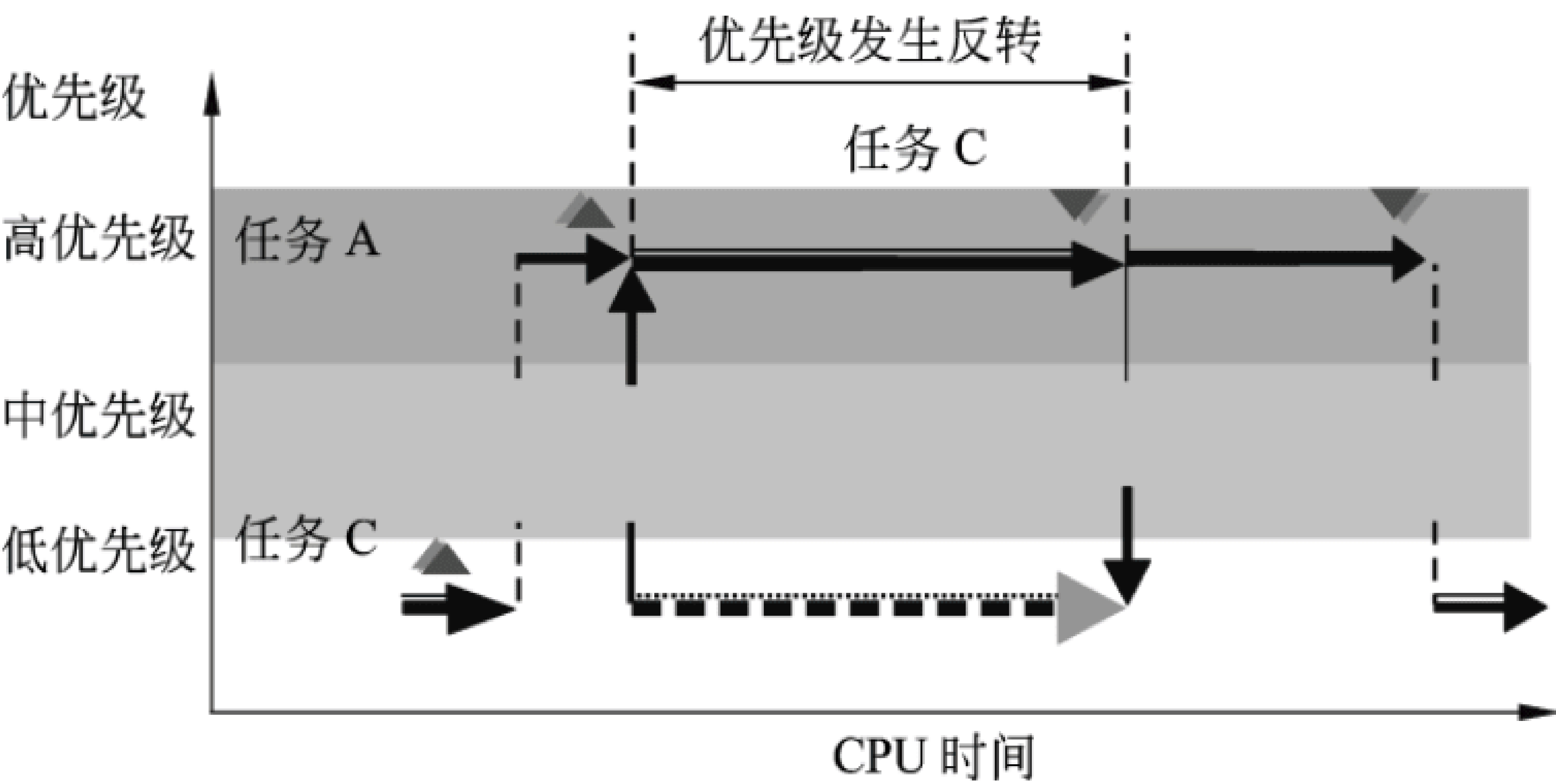


图 9-9　优先级继承的实例

2．天花板优先级协议

如果每个任务的优先级是已知的，对于给定资源，其优先级天花板是所有可能需要该资源的任务中最高的优先级，如表 9-3 所示。

表 9-3 天花板优先级协议规则

协议规则号	描　述
1	如果资源 R 在使用，则任务 T 被阻塞
2	如果资源 R 是空闲的，则资源 R 被分配给任务 T。如果资源 R 的优先级天花板比任务 T 的优先级高，则任务 T 的优先级被提升到资源 R 的优先级天花板等级。在任意给定时间，任务 T 的执行优先级等于所有它占有的资源中最高优先级天花板
3	当具有最高优先级天花板的资源被释放时，任务 T 的优先级分配给另一个资源的次最高优先级天花板
4	当释放所有资源后，任务恢复到原来分配的优先级

使用天花板优先级协议时，一旦某任务获得该资源或暂无其他较高优先级的任务竞争同样资源时，则此任务便继承该资源的优先级天花板。这意味着访问某临界资源的所有任务的临界区具有同样的天花板等级。

3．优先级天花板协议

任意时刻，一个运行系统的当前优先级天花板（Current Priority Ceiling）是此时所有正在使用的资源中的最高优先级天花板。例如，系统中有三个资源正在使用，资源 R1 的优先级天花板为 4，资源 R2 的优先级天花板为 6，资源 R3 的优先级天花板为 9，则系统当前的优先级天花板为 9。

优先级天花板协议具有如下特性：

（1）一个请求任务只可以被一个任务阻塞。

（2）在优先级天花板协议下，不会发生传递阻塞。

（3）在优先级天花板协议下，不会发生死锁，优先级天花板协议规划如表 9-4 所示。

表 9-4 优先级天花板协议规则

协议规则号	描　述
1	如果资源 R 在使用，则任务 T 被阻塞
2	如果资源 R 空闲且任务 T 的优先级比当前优先级的天花板高，则资源 R 分配给任务 T
3	如果当前天花板属于任务 T 当前保持的资源之一，则资源 R 分配给任务 T，反之，任务 T 被阻塞
4	如果阻塞任务 T 的任务的优先级更高，则继承 T 的优先级，并且按此优先级执行，直到它释放每个优先级天花板高于或等于任务 T 的优先级的资源，然后，任务回到先前的优先级

9.6 嵌入式软件移植

由于嵌入式系统的专用性特点，系统的硬件平台和软件平台多种多样，每种都针对

不同的应用而专门设计，因此，应用软件在各个平台之间很少具有通用性，并且嵌入式系统的更新换代速度相对较快。为了保护已有的投资、充分利用现有的软件资源和加快产品研制速度，软件的移植在嵌入式领域变得非常频繁。

一般地，通用计算机软件移植的情况并不频繁，而嵌入式系统软件移植的情况却很多，主要有如下两个原因：

（1）基于嵌入式处理器的原因。每种嵌入式处理器都有一个生命周期，随着半导体器件技术的迅猛发展，嵌入式处理器制造商不断推出新的嵌入式处理器产品，新的产品与旧的产品在一定时期内可能是向下兼容的。但是随着新的高性能体系结构的研究与开发，旧的体系结构可能被淘汰或停产，新的处理器体系结构不一定兼容旧的体系结构，导致以前的软件不能在新的结构上继续运行，于是，需要考虑现有系统的软件移植问题。

（2）基于操作系统的原因。以前，嵌入式系统的开发工具很不完善，大部分嵌入式系统与嵌入式操作系统紧密相关，一种嵌入式应用只能运行在一种嵌入式操作系统之上，如果操作系统的换代、老操作系统的淘汰或因技术或市场的原因而要求应用软件运行在新的操作系统平台上，则应用软件就要进行基于操作系统移植。

嵌入式系统的软件移植可以归纳如下两类：基于裸机系统的软件移植（基于裸机的应用软件移植到新的硬件平台上；嵌入式操作系统移植到硬件平台上）、基于嵌入式操作系统的移植（应用软件移植到新的操作系统之上；操作系统和应用软件的整体移植）。

9.6.1　裸机系统的软件移植

早期，直接在裸机系统上开发的应用软件要移植到新的体系结构不同的嵌入式系统上运行时，这类软件通常比较简单，软件的代码量不大，结构不太复杂，如果大部分代码采用高级语言编写，仅有少部分为汇编代码，这种情况下移植的工作量不大，只需将与系统结构相关的汇编代码重新改写后，就可完成移植。否则，若是整个代码均采用汇编语言编写，移植基本上等于重新开发。

应用软件大部分采用高级语言编写，且具有良好的层次结构，如图 9-10 所示，软件分为两层，输入/输出模块层已嵌入式硬件平台为运行平台，实现了系统 I/O 接口的软件支持，完成设备驱动程序的功能，如设备控制、数据输入/输出等。这类软件的移植只需修改与处理器相关的 I/O 模块即可。

如果在处理器之上又增加了一个硬件抽象层，如图 9-11 所示，则把 I/O 模块设计成不是依赖于处理器体系结构而是基于硬件抽象层的，由硬件抽象层来屏蔽硬件细节的多样性，这种三层结构的软件的移植工作量会进一步减少。

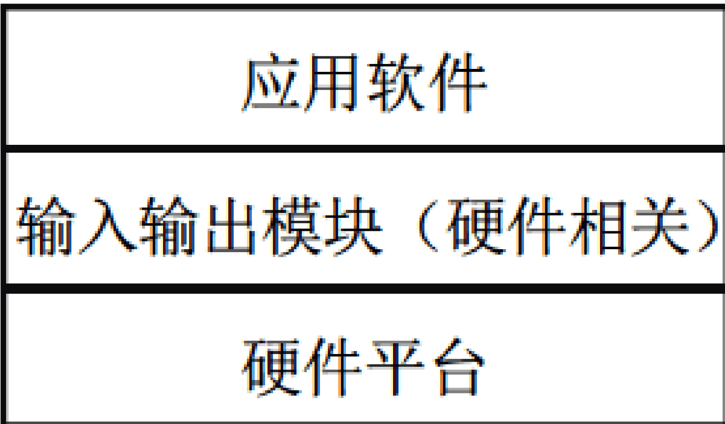


图 9-10　模块化应用软件移植

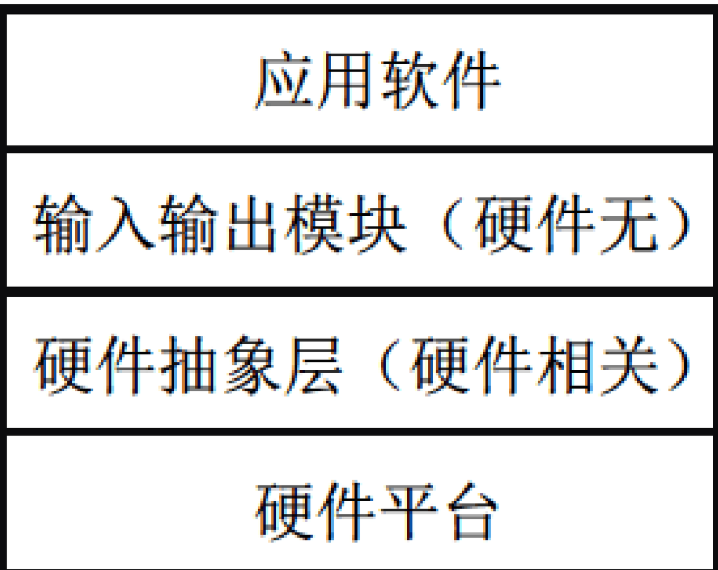


图 9-11　具有硬件抽象层的软件移植

9.6.2　基于操作系统的软件移植

有操作系统支持的嵌入式系统软件系统如图 9-12 所示，这种情况下的移植工作包括操作系统在新的平台上的移植和新平台的设备驱动程序的开发和移植。与移植有关的软件系统包括嵌入式操作系统、设备驱动程序两大部分。

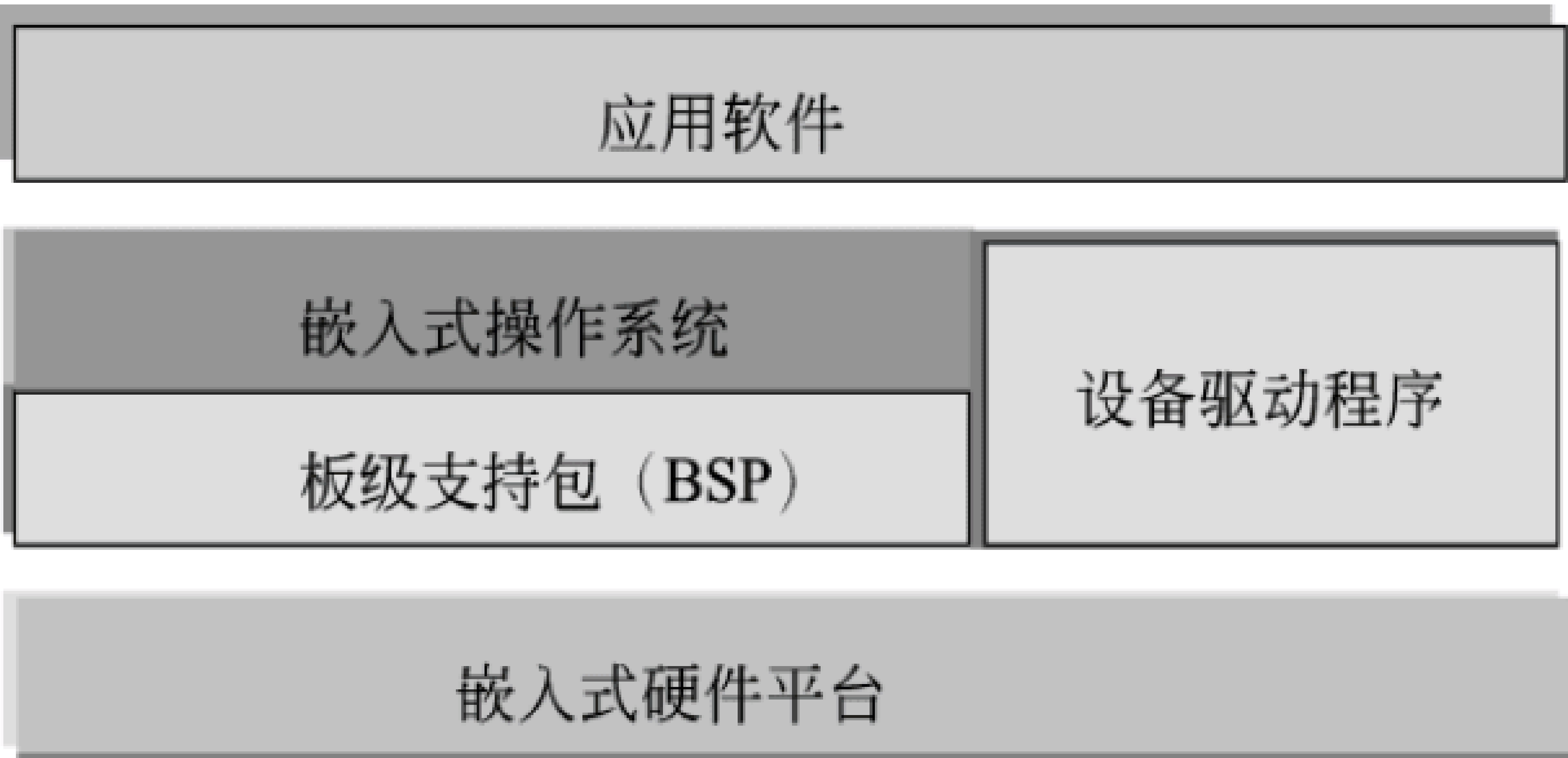


图 9-12　带有嵌入式操作系统支持的软件结构

1．板级支持包

BSP（板级支持包），是对目标系统的底层支持软件。具体地说，BSP 是一些与硬件相关功能的集合，这些文件按功能大致可分为三个部分：

（1）操作系统载体的硬件初始化文件。

（2）操作系统初始化文件。

（3）生成 BSP 目标代码所需的工具文件，如各种编译链接文件等。

BSP 的主要工作有以下一些：

（1）硬件初始化，主要是 CPU 的初始化，为整个软件系统提供底层的硬件支持。

（2）为操作系统提供设备驱动程序和系统中断服务程序。

（3）定制操作系统的功能，为软件系统提供一个实时多任务的运行环境。

（4）初始化操作系统，为操作系统正常运行做好准备。

希赛教育专家提示：在移植操作系统到新的硬件平台上时，BSP 需要用户重新编写，一般嵌入式系统供应商会提供一些编写帮助和示例参考。

2．设备驱动程序

设备驱动程序是设备提供给操作系统或者应用软件的一套接口，主要负责对硬件寄

存器的读/写操作和设备的逻辑控制。它的出现把操作系统和应用软件与设备隔离开来，屏蔽了硬件的细节，方便了用户对设备的读/写和控制。同时它也使得一种硬件设备只要配备不同的驱动程序，就可以在不同的系统上使用。一种操作系统只要配备不同的驱动程序，就可以使用不同设备。

设备驱动程序是内核的一部分，它完成以下功能：

（1）对设备初始化和释放。

（2）把数据从内核传送到硬件和从硬件读取数据。

（3）读取应用程序传送给设备文件的数据和回送应用程序请求的数据。

（4）检测和处理设备出现的错误。

在移植操作系统到新的硬件平台上时，设备驱动程序需要用户重新编写。

9.6.3　层次化设计

在设计嵌入式软件时，设计的早期就考虑到软件系统的移植性问题是一个很好的设计思路和习惯，设计易于移植的嵌入式软件需要遵循层次化和模块化的软件设计方法。

1．可移植性软件的设计原则

可移植软件的设计原则主要是层次化和模块化。

（1）层次化。所谓层次化，是指嵌入式系统的软件设计的纵向结构，下层为上层提供服务，上层利用下层提供的服务完成更高级的功能。下层向上层的服务通过调用接口的形式提供。层次化的嵌入式软件结构体现在不同层次的软件模块的相互依赖关系上，层次化的软件结构给每个层次定义一个清晰的接口和功能，分层的数量要合理，不能太多，以免增加软件的复杂性和降低效率，又不能太少，否则同一层里存在依赖关系，失去了分层的意义。层次化的软件在移植的时候，只需修改底层的相关软件即可，不需要修改上层软件。修改底层软件的时候，保持向上的调用接口不变，上层的软件无需修改即可在新的环境下运行。

（2）模块化。在同一层次上，软件又被划分为一个个单独软件单元，即模块。一般模块之间是相互独立的，一个模块的实现不依赖于同层的其他模块，模块之间的通信也常常不用全局变量，可以使用操作系统的服务机制，良好的模块设计可以使软件易于裁减和更新。

（3）层次化与模块化的结合。规模较大的系统设计，采用模块化和层次化相结合的方法，软件的上、下层之间采用层次化设计，同一层的内部采用模块化的设计。层次化的最低层是硬件抽象层，为嵌入式操作系统的移植提供可移植的环境。

2．硬件抽象层

如图 9-13 所示，硬件抽象层把系统软件和硬件部分隔离开来，这样就使得系统的设备驱动程序与硬件设备无关，从而大大提高了系统的可移植性。从软硬件测试的角度来看，软、硬件的测试工作可分别基于硬件抽象层来完成。在抽象层的定义上，需要规定

统一的软、硬件接口标准。

硬件抽象层及其接口具有如下特点：

（1）硬件抽象层具有与硬件密切的相关性。

（2）硬件抽象层具有与操作系统的无关性。

（3）接口定义的功能应包含硬件支持的所有功能。

（4）接口的定义应简单，以免增加软件的复杂性。

（5）接口的设计应具有可测试性，以利于系统的测试和集成。

实际上，在具体的实现中通常可以将硬件抽象层分为5种类型，即公共抽象层、体系结构抽象层、变体抽象层、平台抽象层和辅助抽象层。

（1）公共抽象层包含了所有体系结构和平台的硬件抽象层所共享的选项和函数，包括常用的调试功能、驱动程序接口函数、监视调用接口等。

（2）体系结构抽象层对处理器的基本结构进行抽象和定义，包括中断和异常的定义和处理、上下文切换程序、Cache 的定义和控制、系统启动程序、测试程序等。

（3）变体抽象层抽象和封装该处理器在处理器系列中所具有的特殊性，包括内存管理部件（MMU）、浮点运算部件、片内设备驱动程序及对体系结构扩展的程序。

（4）平台抽象层对当前的系统的硬件平台进行抽象，包括平台的启动、芯片的选择和配置、定时器、I/O 寄存器访问以及中断寄存器等。

（5）辅助抽象层包含了处理器的一些变体所共享的公共模块和特殊模块。

3．操作系统抽象层

如图 9-14 所示，操作系统抽象层把应用软件和操作系统隔离开来，使得应用软件与特定的操作系统无关，从而大大提高了应用软件的可移植性和跨平台的支持性。



图 9-13　包含硬件抽象层的系统结构

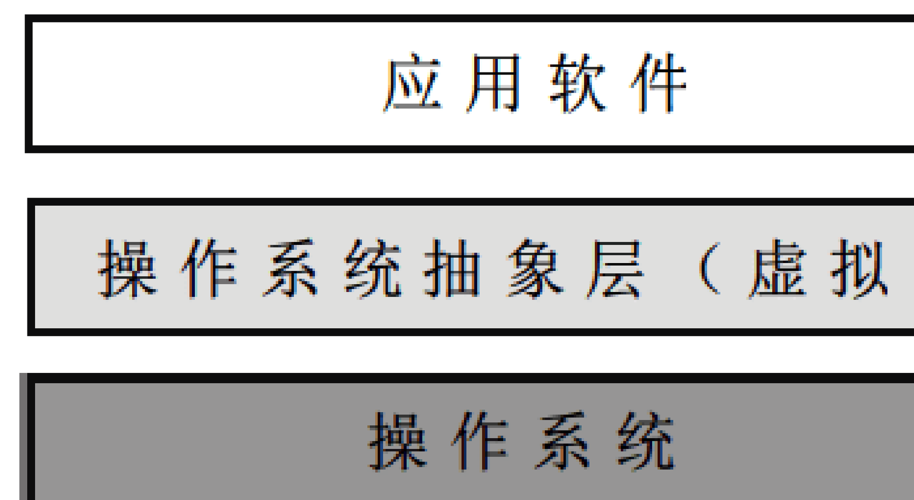


图 9-14　含有操作系统抽象层的软件结构

在操作系统抽象层的定义方面，需要规定统一的操作系统接口标准，这些工作根据系统的需要和操作系统的多样性来进行。完善的嵌入式操作系统抽象层的功能应该覆盖操作系统的所有功能。

操作系统抽象层为应用软件的开发提供了一个一致的编程接口，包含的功能有任务的管理、任务之间的通信、常见的操作系统服务如内存管理、中断和异常处理等。

操作系统抽象层的主要特点有以下一些：

（1）操作系统抽象层跟实际操作系统密切相关，统一的操作系统接口标准是在实际操作系统的编程接口的基础上封装并扩充而来。

（2）操作系统抽象层跟应用软件无关，应用软件只需使用统一接口标准编写即可。

（3）操作系统抽象层的实现本身应该可裁减。

（4）接口标准应该简单统一，应尽量支持 POSIX 标准。

（5）接口设计应该具有可测试性，以利于系统的集成与测试。

（6）操作系统抽象层的实现应尽量保持原来系统的性能。

本章参考文献

[1] 刘斌，高小鹏．嵌入式软件可靠性仿真测试系统研究．北京航空航天大学学报，2000（8）：490-493

[2] 杨水清，张剑，施云飞. ARM 嵌入式 LINUX 系统开发技术详解．北京：电子工业出版社，2008

[3] 何宗键. Windows CE 嵌入式系统．北京：北京航空航天大学出版社，2006

[4] 陈渝，韩超，康烁．嵌入式系统实践教程．北京：机械工业出版社，2008

[5] 沈建华. ARM 嵌入式系统开发：软件设计与优化．北京：北京航空航天大学出版社，2005

[6] 杨刚，龙海燕．嵌入式系统设计与实践．北京：北京航空航天大学出版社，2009

[7] 马洪兵，谷源涛．嵌入式系统：硬件与软件架构．北京：人民邮电出版社，2008

[8] 宋劲杉，王华勇，邸海霞．嵌入式 LINUX 系统设计与开发．北京：电子工业出版社，2008

[9] 康一梅．嵌入式软件设计．北京：机械工业出版社，2007

[10] 陈曙晖，王继进．嵌入式系统——体系结构、编程与设计．北京：清华大学出版社，2005

[11] 刘洪涛，孙天泽．嵌入式系统技术与设计．北京：人民邮电出版社，2009

[12] 蓝枫叶．嵌入式操作系统设计与实现．北京：电子工业出版社，2008

[13] 俞建新，王健，宋健建．嵌入式系统基础教程．北京：机械工业出版社，2008

[14] 怯肇乾．嵌入式系统硬件体系设计．北京：北京航空航天大学出版社，2007

第 10 章　文 档 编 制

软件文档是软件产品的重要组成部分，对于开发人员、管理人员以及用户都是十分重要的辅助工件。定义清晰、维护及时的文档能够帮助开发人员理解需求、顺畅沟通；帮助管理人员了解进度、加强管理；帮助用户更好使用和维护软件。因此，对于系统分析师而言，必须掌握软件文档编制的技能。

根据考试大纲的要求，在文档编制方面，根据《GB 8567—2006 计算机软件产品开发文件编制指南》标准，要求考生掌握可行性研究报告、项目开发计划、需求规格说明书、数据要求规格说明书、用户手册、操作手册、测试计划、测试分析报告、技术报告、开发进度记录、项目开发总结报告等编写方法。

10.1　软件文档概述

在软件的开发过程中，清晰、正确、规范的软件文档将起到十分重要的作用，主要体现在以下几个方面：

（1）可以有效提高软件开发过程的可视度，满足用户、管理人员和开发人员等不同视角的需求。

（2）通过编制文档，能够更早地发现错误，提高开发效率。

（3）可以作为开发过程中的工作成果和阶段结束标志，便于里程碑式的项目管理。

（4）详尽的信息记录，能够为软件开发、使用和维护工作提供支持。

（5）能够促进软件开发过程的规范化。

（6）能够为团队建立经验模型和可复用库，提高团队产能。

不过，值得注意的是，不要过于强调文档，毕竟文档并不是“可运行的成果”，最有价值的成果还是最后的“可执行系统”。很常见的滥用文档的现象是团队不断地追求文档的厚度、完整性，甚至是花很大的时间去美化文档、不断地更新并不重要的文档，这样不仅不会对软件开发带来好处，反而会对开发效率带来很大的害处。

大家都知道，CMM、UP 等重量级方法和 XP、FDD 等轻量级方法（也称为敏捷方法）对于“量”的定义就是“文档量”，从表面上看，一方的观点是通过大量的文档来规范过程；另一方的观点则是抛弃繁文缛节的文档，来提高过程的灵活性。

在笔者的实践中，得出一个道理，关于文档的量的大小，是一个度的问题。重量级方法是针对长期以来 Code-fix 的开发模式提出的，以规范化的过程，并辅之详尽的文档来弥补其缺陷。但如果对于一个高效的团队，文档过多将引起反效果，就需要进行精减，

以提高团队的灵活性，这也是一种辩证的思维。

希赛教育专家提示：综合来说，在软件开发过程中，应该充分注重文档的实效，而非形式，形而上学的文档编制工作对于软件开发来说将带来新的问题。

1．文档的分类

根据软件文档产生、使用的范围的不同，可以将其分为三大类。

（1）开发文档：为开发工作提供支持的各种文档，其读者群主要针对开发人员。其中主要包括需求规格说明书、数据要求规格说明书、高层设计说明书、详细设计说明书、项目开发计划等。

（2）管理文档：为项目的开发管理提供支持的各种文档，其读者群主要针对管理人员，其中主要包括可行性研究报告、项目开发计划、测试计划、技术报告、开发进度记录、项目开发总结报告等。

（3）用户文档：向用户传达各种与开发相关、与产品相关的信息，其读者群主要针对最终用户。其中主要包括用户手册、操作手册、维护修改建议书、软件需求说明书等。

在编写软件文档时，可以采用自然语言直接编写，也可以采用形式化语言来编写，还可以借助各种图表提高可读性，例如 UML、DFD、E-R 图等。

2．文档编制的要求

要有效地发挥文档在软件开发过程中的作用，必须确保文档的高质量。而高质量的文档包括以下几个主要特点。

（1）针对性：文档编制时需要根据面向的读者选择描述的手段，例如针对开发人员就应该尽量使用形式化语言，采用专业的术语、图表；针对用户则应该尽量使用自然语言，使用用户领域的术语。

（2）精确性：文档中的描述必须做到无二义性，否则不同的读者阅读相同的文档却产生了不同的理解，将带来很大的麻烦。

（3）清晰性：文档应该尽量做到简明，尽量采用图表等直观的形式进行说明，以保证其清晰易懂。

（4）完整性：每一个文档都应该自成体系，不要过多的互相依赖，以避免造成要理解一个问题，需要在多个文档中来回翻看。

（5）灵活性：虽然针对每种文档，有许多可以借鉴的模块，但是在编制的时候不可形而上学地生搬硬套，应该根据项目的规模、复杂程度等因素进行适当和必要的剪裁，灵活地处理。

值得一提的是，所有的文档主要是回答 5W1H 的问题，即 What（做什么）、信息从何处来（Where）、何时做（When）、由谁来做（Who）、怎样干（How）、为什么要进行开发（Why）。

3．文档的管理与分发

在整个软件生存周期中，软件文档会不断地产生、更新，因此需要对文档进行必要

的管理。具体来说，主要包括以下几个关键环节。

（1）文档的产生：首先应该根据企业、项目的实际情况，明确项目中应该编制的文档内容，并且为文档编写相应的编制指南，确定编制人/小组，并入项目计划作为里程碑的检查点。另外，还应该建立相应的评审机制，以保证文件符合要求。

（2）文档的标识：为了能够更好地管理文档，应该对每一个文件进行标识，制定文档号的规范（通常包括项目名、种类、版本等内容）。

（3）文档的版本控制：由于在软件开发过程中，文档也与代码一样，会不断地发生变化，因此将文档纳入配置管理范围是一个好办法，而且通过配置管理系统也能够良好地构建起一个分发机制。

（4）文档的变更及分发管理：为了保证文档与代码的匹配性，需要制定相应的变更管理、分发管理的制度。我们可以利用配置管理系统的权限管理功能，保证文档只被允许的人员阅读、修改。

10.2 可行性研究报告

《可行性研究报告》是可行性研究活动的产物。编制《可行性研究报告》的目的说明该软件开发项目在经济、技术、社会等方面的可行性，阐述能够达到开发目标的各种合理解决方案，并提出建议选择的解决方案以及相应的理由。

在许多开发组织中，并未对可行性研究活动产生重视，毕竟很多项目都是因客户的订单而产生的，因此，也就很少编制《可行性研究报告》。但是，对于一个追求成功的开发组织来说，可行性研究活动是不应该省略的，一个不可行的项目，不管团队花费多大的努力，终究难逃失败的宿命。

在 GB 8567—2006 标准中，提供了一个《可行性研究报告》的文档模板和编写指南，其中规定了在可行性研究报告中应该包括的内容。

1．引言

说明文档的编写目的和预期读者；为文档阅读者提供一些项目相关的背景资料；对项目中所使用的术语和缩写进行解释；列出所有用得着的参考资料。这部分的内容也是绝大部分文档相同的地方。

2．可行性研究的前提

要进行项目的可行性研究，就必须对要研究提供一些基础素材，包括项目的要求、目标、假设条件以及相关的设计约束。

（1）要求：主要包括功能要求（如果可能，可以对系统的输入、输出以及处理进行描述）、性能要求、安全性系统、相关联的外部系统以及完成期限。

（2）目标：要求是要项目达到什么功能？而目标则是通过项目获得什么？例如提高企业资金周转率、缩短业务处理周期等。

（3）假设条件及相关的设计约束：包括一些资金、政策、软硬件资源等方面的假设条件或约束条件。

另外，在本部分中还应该说明将采用的可行性研究方法，如调查、加权、确定模型、建立基准点或仿真等。以及评价系统所使用的主要尺度，如费用的多少、各项功能的优先次序、开发时间的长短、使用的难易程序。

3．对现有系统的分析

这也是可行性分析中的一个重要组成部分，这个现有系统可能是一个老的计算机系统，也可能是一个人工信息，即非信息化系统，通过这方面的分析可以引出开发该系统的意义。这个部分主要包括以下几个方面的内容。

（1）处理流程和数据流程：常用图表、流程图的形式说明。

（2）工作负荷：工作及工作量的说明。

（3）费用开支：运行原系统所引发的人力、设备、空间、服务、材料等各种费用。

（4）人员、商务：原系统所需人员和设备情况。

（5）局限性：这就是要开发新系统的主要原因与意义。

4．所建议的系统

本部分主要是针对系统的目标和要求，提出一个可行的解决方案。并且针对这些因素，论证系统是如何满足的。

（1）对所建议的系统的说明：说明解决方案是如何满足在第二部分中列出的要求的，以及提供相关的方法和理论根据。

（2）处理流程和数据流程：新系统的各方面流程，通常用图表、流程图表示。

（3）改进之处：针对原系统的局限性进行说明，列举出该系统所改进的内容。

（4）影响：说明新系统对于设备、软件、用户单位机构、系统运行过程、开发、地点和设施、经费开支等方面的影响。这些影响应该包括正面的和负面的，主要是负面的。

（5）局限性：说明新系统仍然存在的不足，以及这些问题未能消除的主要原因。

（6）技术条件方面的可行性：在现有技术、当前限制条件下，是否有足够的开发人员在规定的期限内完成系统开发。

5．可选择的其他系统方案

在第 4 部分中是建议选择的解决方案，而在这一部分中则应该列出其他所有可以选择的解决方案。每一个方案的描述可以按照第 4 部分的内容进行描述，另外还应该逐一说明其未选中的理由。

6．投资及效益分析

这部分内容主要是从经济角度来说明解决方案的可行性，主要包括以下内容。

（1）支出：也就是投资的说明，应该包括基本建设投资、其他一次性支出、非一次性支出三方面的内容。

（2）收益：也就是说明该系统能够带来的收益，可以是直接收益，也可以是节省的

成本。应该包括一次性收益、非一次性收益、不可定量的收益三方面的内容。

（3）收益/投资比：计算出整个系统生命期内的收益/投资比值。

（4）投资回收周期：计算出累计收益超过累计支出的时间。

（5）敏感性分析：指对一些关键要求变化时对成本和收益的影响分析。

7．社会因素方面的可行性

这部分内容主要是从社会角度来说明解决方案的可行性，主要包括是否符合法律法规，在机构设置和员工素质方面是否能够保障系统可用。

8．结论

《可行性分析报告》中最重要的是要给出一个鲜明的结论，是可立即执行，还是要等某种条件满足以才可执行，或者是需要对目标进行某些修改后才能够开始进行，当然如果不可行，那么也应该直接说明其不能进行或不必进行的结论。

《可行性研究报告》应该在立项前完成。对于项目型软件，《可行性研究报告》的初稿应该是在客户方完成的，但开发团队应该在承接项目之后，指定专门的人员协助客户一同进行完善，从而明确可行的解决方案。对于产品型软件，则应该由市场部门、开发部门共同协作，从而明确可行的解决方案。

参与编写《可行性研究报告》的人员，应该包括系统分析人员（对计算机软件技术有着广泛的理论与实践知识基础）、资深系统开发人员（对具体的实现技术有着良好的经验）、客户代表以及相关的法律、市场顾问。

10.3 项目开发计划

《项目开发计划》是对开发过程中承担各项工作的人员、预计的进度、所需的经费，以及所需计算机软、硬件资源等方面的问题做出安排的重要文档，是项目管理与监控的基本依据。

在实践中，项目开发计划的编制却经常流于形式，一旦编制完成便束之高阁，再也不用。究其原因，发现主要是在软件开发过程中，变化极快，往往彻底打破了原计划，因此团队根本不相信项目开发计划。如此日久天长，业内便流传着“计划计划，全是鬼话”的戏语。要解释其中道理，在此还需借用著名的军事家拿破仑的一句经典名言：“没有一场战役是按计划进行的，但也没有一场没有计划的战役”。其辩证地说明了计划的两面性，一方面事件的发展是充满变数的，是无法预先获知的；另一方面根据变化动态的制定计划是成功的保证。

在 GB 8567—2006 标准中，提供了一个《项目开发计划》的文档模板和编写指南，其中规定在项目开发计划中除了引言部分之外，应该包括以下几个部分内容。

1．项目概述

该部分的内容主要是对要进行的项目进行一个概要性描述，以使团队内每一个成员

对项目有一个总体的了解，通常包括以下内容。

（1）工作内容：必须完成的各项工作。

（2）主要参与人员：项目组成员，以及专业技能情况说明。

（3）产品：包括需要交付给客户的程序、各类相关文件以及配套的服务，以及其他一些团队内部留存的源码等其他工件。

（4）验收标准：这是本部分中最关键的一节，其主要用来帮助开发人员更清晰地了解要求，帮助客户更好地验收产品。

（5）完成项目的最迟时间：也就是相应的时间限制。

（6）计划的批准者和批准日期。

2．实施计划

该部分则是对项目的实施进行详细的安排与计划，其中包括以下内容。

（1）工作任务的分解与人员分工：按软件生命周期对工作任务进行按阶段分解，建议采用 WBS 方法。然后针对分解生成的每项任务指定专门的负责人，职责到人。

（2）接口人员：与该项目相关的项目干系人，包括用户、本单位的管理部门、外包方代表等。

（3）进度：对项目实施的进度进行安排，通常采用甘特图或 PERT 图表示。

（4）预算：列出该实施计划所需的经费安排，包括人员工资、办公费、差旅费、机时费、资料费、通信设备和专用设备的租金等。

（5）关键问题：列举影响项目成败的关键因素，以及它们的影响及针对性措施。

3．支持条件

列举该项目开发所需要的各种配套的条件和设施，其中主要包括计算机软、硬件以及专用设备资源；用户所需承担的配合工作；外单位提供的条件等。

4．专题计划要点

对于项目开发中的配套过程进行计划，通常包括分合同计划、开发人员培训计划、测试计划、安全保密计划、质量保证计划、配置管理计划、用户培训计划、系统安装计划等。其编写的详略程序根据项目的大小决定，针对较大的项目可以考虑将其中的一些内容作为专项计划。

《项目开发计划》是一个逐渐求精的过程。通常在项目的可行性分析之后，项目开发计划的初稿就应该形成，这时的计划应该是粗略的，其只是针对“工作结构分解”中的较高层进行初步的进度估算、资源安排。得出来的各种计划值应该是一个区域值，如 3～5 人月，而不应该是一个精确值。

接着就是随着项目的迭代，逐步的精化。每一次迭代，就会将工作任务切出一个小块，然后对这个小块进行准确的估算，做出相对精确的估算。并且在执行的过程中，根据实际的进度进行动态的调整。

在大多数的项目中，项目计划是由项目经理一手炮制的，甚至有可能是来自市场部

门、管理层的直接压力，在不可能的最后期限的限制下，编造出来。这样的项目计划就会失去其意义，因为其是脱离实际建立的。正确的方法是，以项目经理为主，与团队成员一起编制，让开发人员一同进行估算，自下而上，形成一个真正的可操作的项目开发计划。

项目经理需要具备估算与度量方面的技能，而且应该能够对开发团队进行培训，以使得他们都能够良好地配合。同时，也会发现以往项目的历史数据，对于度量来说有着多么重要的参考作用。

10.4 需求规格说明书

《需求规格说明书》是需求分析活动的产物，编制该文档的目的是使项目干系人（客户、用户）与软件开发团队双方对该软件的初始规定有一个共同的理解，使之成为整个开发工作的基础。《需求规格说明书》是软件开发过程中最重要的文档之一，对于任何规模、性质的软件项目都不应该缺少。

在需求过程中，分析人员可以通过用户访谈、用户调查、文档研究、现场观摩、JAD等需求获取手段获取零散的需求信息；然后通过用例建模、结构化分析、信息工程等手段进行分析、总结，最后将其归并为《需求规格说明书》。由于该文档是整个软件项目的最关键文档之一，因此写作时其完整性、正确性、可行性是十分重要的。而且为了能够让非技术人员更好地阅读和理解，应该尽可能通过自然语言、简单的图表来表达，以防止造成不必要的误会。

在 GB 8567—2006 标准中，提供了一个《软件需求规格说明书》的文档模板和编写指南，其中规定在需求规格说明书中除了引言部分之外，应该包括以下几个部分内容。

1．任务概述

该部分是对软件系统要完成的任务做一个总体性的描述，让项目干系人在较高的层面上达成共同理解，通常从以下三个角度进行描述。

（1）目标：说明该软件的开发意图、应用目标、作用范围等，如果该系统与其他系统有很强的关联性，则可以采用一些图表（如工作上下文范围图）来表示。

（2）用户的特点：列举本软件的各类最终用户，并说明这些用户的教育水平、计算机使用水平、技术专长以及今后使用软件的频度等，这些内容将作为软件设计工作的重要约束条件。

（3）假定和约束：一些如经费、开发时间、系统选型等方面的假定或约束条件。

2．需求规定

在本部分将详细地对需求进行规格化定义，让用户与开发人员能够在具体需求细节方面达成共识，通常包括以下几个方面。

（1）对功能的规定：通过 IPO（输入、处理、输出）表的形式，逐项定量、定性地

叙述软件所需的功能要求。

（2）对性能的要求：包括精度、时间特性、灵活性等非功能要求。

（3）输入输出要求：解释各个输入、输出数据的类型、媒介、格式、数值范围、精度等。

（4）数据管理能力的要求：说明该系统将可能产生、存储与处理数据的规模。

（5）故障处理要求：列出所有可能出现的软、硬件故障，以及说明发生的后果和应如何进行处理。

（6）其他专门要求：包括安全性、可维护性、可扩展性、易读性、可靠性、运行环境等其他方面的需求。

3．运行环境规定

本部分主要用来描述系统对运行环境的要求，通常包括设备、支持软件、接口（通信协议）、控制信号等方面的要求。

《需求规格说明书》是需求分析完成之后，进行规格化说明的产物。在需求过程中通常会经历三个阶段：需求捕获、需求分析、需求规格化。需求捕获阶段将从用户处获得大量的零散的需求信息；而需求分析阶段将对这些零散的需求信息进行整理，生成分析模型；而在需求规格化阶段则主要是在分析模型的基础上，整理成为规格化的说明书，建立需求基线。

需求捕获、分析与规格化的工作通常就是由系统分析师来承担，要求这些人员能够有良好的沟通能力、写作能力，应该具有站在客户的角度看系统的思维逻辑，并且需要有效地掌握各种需求捕获技术、分析技术以及对计算机技术知识有着广博的知识。

10.5　数据要求规格说明书

如果项目中有大量的各种数据需要处理，而且这些数据的采集、加工对系统来说有着十分重要的作用，或者是有大量的原始数据需要导入到新的系统，那么就有必要对这些数据要求进行分析，并编制《数据要求规格说明书》。数据要求说明书能够向整个开发时期提供关于被处理数据的描述和数据采集要求的技术信息。

在 GB 8567—2006 标准中，提供了一个《数据要求规格说明书》的文档模板和编写指南，其中规定在数据要求规格说明书中除了引言部分之外，应该包括以下几个部分内容。

1．数据的逻辑描述

根据数据的逻辑属性不同，可以分为动态数据和静态数据两种。动态数据就是在运行时会经常变化的，而静态数据则是在运行时很长一段时间不会变化的，一般也随着系统的运转而发生改变。对于每个数据元，应给出名称、定义、度量单位、值域、格式和类型等信息。具体来说，可以分成静态数据、动态输入数据、动态输出数据、内部生成

数据、数据约定 5 个方面进行描述。

2．数据的采集

该部分则是具体展开说明数据如何获取，主要包括以下内容。

（1）要求和范围：根据数据元的逻辑分组进行说明数据采集的要求和范围，说明具体的采集方法，其中的内容通常包括：

① 输入数据的来源：例如操作员、数据输入工作站、专业数据输入公司等；

② 数据输入所用的媒介或设备；

③ 说明输出数据的接受者；

④ 输出数据的形式或设备；

⑤ 数据值的范围；

⑥ 数据的度量单位；

⑦ 处理的频度。

（2）数据输入的承担者：说明输入工作将由谁完成，用户或开发者，或者是某种设备，也可以是某个接口软件。

（3）预处理：说明数据采集后的一些相应的预处理工作，包括数据格式转换等。

（4）影响：说明这些数据要求对于设备、软件、用户、开发单位所可能产生的影响。

10.6 用户手册

“只有当客户产生了价值，那么软件系统才能够获得价值”，因此，提供有效的支持，帮助用户能够更好地使用软件系统，最大限度地发挥出其潜在作用，是软件项目开发中的一个重要环节。而编制《用户手册》、《操作手册》工作，正是为客户提供支持的有效手段，不过它们是用来解决不同的问题的，在内容与组织方面都有很大的区别。下面首先还看看《用户手册》的用途。

《用户手册》是使用接近客户的自然语言（避免使用专业术语），充分地描述该软件系统所具有的功能及基本的使用方法。使用户（或潜在用户）通过本手册能够了解该软件的用途，并且能够确定在什么情况下，如何使用它。

不过，对于较小型的软件系统而言，用户手册和操作手册可以合二为一，其中的内容可在《用户手册》和《操作手册》编写指南的基础上进行修改与整合。

在 GB 8567—2006 标准中，提供了一个《用户手册》的文档模板和编写指南，其中规定在用户手册中除了引言部分之外，应该包括以下几个部分内容。

1．用途

该部分主要对整个软件系统进行概要性说明，让读者了解该软件的功能以及适用于什么情况，通常从以下几个方面进行描述。

（1）功能：结合软件的开发目的，逐项地说明该软件系统所具有的各种功能以及它

们的适用范围。

（2）性能：说明软件的精度（数据的输入、输出及传输中的精度）、时间特性（定量说明响应时间、更新处理时间、数据传输及转换时间、计算时间等）、灵活性（当用户需求变化时软件的适应能力）等关键性能指标。

（3）安全保密：说明该软件在安全保密方面的设计考虑，以及实际能够达到的水平。

2．运行环境

说明运行该软件系统所需的软、硬件及相关环境的要求，是用户手册的一个重要组成部分。通常包括以下几个方面。

（1）硬件环境：列出运行本软件所要求的主机、网络、存储器等硬件环境的要求，这个部分应该尽可能地列出细节，以使用户能够正确地采购和部署相应的设备。

（2）支持软件：说明运行本软件所需要的支持软件，包括操作系统（名称、版本）、程序语言的编译系统名称版本号、数据库管理系统、中间件、第三方支持构件等。

（3）数据结构：说明本软件运行所需要的数据库或数据文件的格式与内容。

3．使用过程

这一部分首先用图表的形式说明软件的功能同系统输入、输出源之间的关系。并对具体的使用操作进行概要性的描述。

（1）安装和初始化：一步步地说明该软件的安装及初始化工作。

（2）输入：详细描述输入数据的操作和相关说明，主要包括以下内容。

① 输入数据的现实背景：输入数据的背景情况（如人员变动）、情况出现的频度、来源、媒介、限制、质量管理等方面的信息。

② 输入格式：长度、格式基准、标号、顺序、标点、词汇表、省略和重复以及控制信息。

③ 输入举例：再长的描述，也不如一个实例更有效，因此最好的办法就是为每个完整的输入形式提供样本。

（3）输出：详细描述输出数据的操作和相关说明。主要包括以下内容。

① 输出数据的现实背景：如谁使用、使用频度、媒介、质量管理等。

② 输出格式：对每一类输出信息进行解释。

③ 输出举例：对每种输出类型提供一个实例。

4．文卷查询

如果软件系统包括有查询功能，那么，就需要描述与数据库查询有关的初始化、准备以及处理相关的详细规定。另外还需要说明查询的功能、方式以及命令的使用说明。

5．出错处理和恢复

为了使得软件的可操作性和可维护性更强，应该对可能出现的错误现象进行说明，通常使用错误代码表来描述，并指导用户在出错时如何进行修改，以及如何恢复。

6．终端操作

如果软件是在基于多终端系统上工作时，应说明终端的配置安排、连接步骤、数据和参数输入步骤、控制规定。说明通过终端操作进行查询、检索、修改数据文卷的能力、语言、过程以及辅助性程序等。

编写用户手册的关键在于，让读者对软件有一个系统的整体的认识。因此在编写使用过程时，应该从用户能够通过软件实现的每一个功能的角度进行整理，而不是按屏幕、系统的外观进行整理。这样编写的好处将使得读者能够很快知道软件都包括什么功能，以及如何使用这些功能。从而帮助用户更好地掌握软件。

用例建模完成之后，就可以开始着手编制用户手册了，然后根据开发的实际情况进行修改。也就是说，用户手册的编写工作应该在需求规格化之后就开始。

《用户手册》的编写工作可以由系统分析师、开发人员或专门的文档支持人员担任，不管是谁，都需要明确地认识到《用户手册》的主要读者是用户，必须采用用户能够理解的语言来编写。

10.7 操作手册

操作手册的编制是为了向操作人员提供该软件每一个运行的具体过程和有关知识，包括操作方法的细节。可以这么说，用户手册相对来说站在比较宏观的角度来向用户介绍软件，而操作手册就是直接从微观、细节中帮助用户使用、操作本软件。

在 GB 8567—2006 标准中，提供了一个《操作手册》的文档模板和编写指南，其中规定在操作手册中除了引言部分之外，应该包括以下几个部分内容。

1．软件综述

本部分主要是对软件系统的结构组成进行简要的说明，通常包括以下内容。

（1）软件的结构：结合软件系统所具有的功能提供该软件的总体结构图表。

（2）程序表：每个程序的标识符、编号和助记名，也就是可执行文件列表。

（3）文卷表：列出其他相关文件的列表。

2．安装和初始化

这部分内容在《用户手册》中也将其包括在内，都是一步一步地说明为使用本软件而需要进行的安装与初始化工作。不过其间的主要区别在于，《操作手册》面向具体的操作者，通常应更加详细。

3．运行说明

运行就是指软件在运行状态下的每一个操作，例如“点击某个菜单项开始，到完成这次操作结束”。这部分的内容可以组织成为以下结果。

（1）运行表：列出各种可能的运行，简略地说明每个运行的目的，以及其所执行的命令。

（2）运行步骤：说明从一个运行到另一个运行以完成系统运行的步骤，其整理的方式应参照《用户手册》中的使用过程。

（3）每个具体的运行说明：包括运行控制、操作信息（目的、操作要求、启动办法、预计完成时间）、输入及输出、恢复过程等。

4．非常规过程

提供有关应急操作或非常规操作的必要信息，如出错处理操作、向后备系统的切换操作以及其他必须向程序维护人员交待的事项和步骤。

5．远程操作

如果本软件能够通过远程终端控制运行，则在本章说明通过远程终端运行本软件的操作过程。

《操作手册》所要达到的目的，通常可以采用“联机帮助”实现，而且联机帮助的效果更佳。由于《操作手册》中主要的描述手段，包括了截屏、图示等，这些主要是想告诉用户具体操作的细节。因此通常阅读操作手册的时候都是在具体的操作时，将帮助文件电子化，用户使用起来更加方便。

另外，还可以采用一些多媒体的手段，制作成为操作演示光盘，其效果会更佳。近年来，还出现了“上下文帮助”技术，可以在具体的操作过程中，随时随刻地向用户提供帮助与支持。

综上所述，设计良好、制作精良的联机帮助系统是《操作手册》最佳的替代品。

由于《操作手册》与用户界面的相关性很大，因此通常会在软件开发完成的时候才开始编制的。如果项目采用的是迭代的开发过程，那么可以在每一次迭代完成，生成相关的可执行程序后，就编写相应的《操作手册》。编制《操作手册》的人员需要有足够的耐心，能够照顾到最初级的操作者，而且还应该具备一定的美工处理能力。

10.8　测试计划

由于测试工作繁杂而且细致，因此，要想获得良好的测试，就必须在测试之前编制相应的计划，将每项测试活动的内容（测试用例）、进度安排、设计考虑、测试数据的整理方法以及评价准则制定出来。

在 GB 8567—2006 标准中，提供了一个《测试计划》的文档模板和编写指南，其中规定在测试计划中除了引言部分之外，应该包括以下几个部分内容。

1．测试计划

对将要进行的测试活动进行详细的说明，其中主要包括以下几个方面。

（1）软件说明：提供一份逐项说明待测试软件的功能、输入及输出、质量指标的图表，这也是测试计划说明的提纲。

（2）测试内容：列出系统测试和验收测试中的每一项测试内容的名称、标识符、进

度安排、内容和目的。

希赛教育专家提示：这里所指的每一项测试内容其实就是一个测试用例集，用于测试其中的一个方面。

（3）针对每一项测试的具体说明，包括进度安排、条件（设备、相关软件及模块、人员）、测试资料、测试培训等方面的内容。

2．测试设计说明

在这个部分说明每一项测试内容的设计考虑，主要包括以下内容。

（1）控制：如输入方法（人工、半自动、自动）、操作的顺序以及结果的记录方法。

（2）输入：测试中所使用的输入数据及选择这些输入数据的策略。

（3）输出：预期的输出数据，如测试结果以及可能产生的中间结果及运行信息。

（4）过程：完成此项测试的步骤和控制命令，即如何完成测试。

其实这里每一个输入、输出及过程都是针对一个测试用例集中的具体测试用例。

3．评价准则

该部分主要是用来描述每个测试的评价准则，其中主要包括以下内容。

（1）范围：说明所选择的测试用例能够检查的范围以及其局限性。

（2）数据整理：说明如果对测试数据进行加工整理，以便于评价。

（3）尺度：说明用来判断测试工作是否通过的评价标准，例如合理的输出结果类型、允许偏差的范围、允许中断或停机的最大次数。

在编制测试计划时，需要相应的人员与专职的测试人员共同参与。而专职的测试人员应该在测试理论基础、测试实践能力、测试工具的掌握方面有一定能力。

10.9 测试分析报告编制指南

只有测试计划，没有分析报告将是不完整的测试，也无法充分发挥测试的作用。在GB 8567—2006标准中，还提供了一个《测试分析报告》的文档模板和编写指南，其中规定在测试分析报告中除了引言部分之外，应该包括以下几个部分内容。

1．测试概要

在该部分建议用表格的形式列出每一项测试的标识符以及相关的测试内容，并指明实际进行的测试工作内容与测试计划的预先设计的结果之间的差别，并说明造成这种差异的相应原因。

2．测试结果及发现

本部分主要是针对每一个测试用例说明测试的结果，以及分析总结其原因。可以分小节对每一个测试的实际得到的动态输出结果，陈述发现的问题。

3．对软件功能的结论

按软件的功能为主线条，对软件的测试结果进行一个总结。对于每一个软件功能做

出以下两方面的描述。

（1）能力：说明为满足此项功能而设计的软件能力，以及经过测试已证实的能力。

（2）限制：说明测试数据值的范围，以及在测试其间发现的缺陷与局限性。

4．分析摘要

本部分是对软件测试完成后，进行一个总体性的分析评价，主要包括以下内容。

（1）能力：经测试证实的软件能力，另外也需要描述测试环境与实际环境之间可能存在的差异性对测试带来的影响。

（2）缺陷与限制：说明经测试证实的软件缺陷和限制，并说明这些缺陷对于软件可能带来的影响。

（3）建议：针对每项缺陷提出改进建议，包括建议的修改方法、紧迫程度、预计的工作量、修改的负责人等。

（4）评价：给软件一个定性的最后评价，即开发是否达到预定目标，是否可以交付使用等等。

5．测试资源消耗

该部分的内容将是对测试工作中使用的资源情况进行一个补充性描述。

10.10　技术报告

作为项目的主要技术骨干，系统分析师经常需要编制各种体裁的《技术报告》，这些技术报告本身就是一个总结经验、共享技术研究成果，实现团队知识共享的有效手段，同时也经常成为项目向上级相关主管部门申报成果的有效途径。

编制《技术报告》的主要目的是将项目开发中技术研究工作做一个总结性的汇报，以文档化的结果将其中的所得保存起来。通常包括以下几个部分。

1．前言

对该文档进行概要性的描述，说明该文档的编写目的、项目背景，并且对其中所使用的专业术语、缩略语建立专门的词汇解释表。

2．本项目解决的技术问题

本部分主要是针对该项目中解决了的技术问题进行一个描述，通常可以分为以下两部分。

（1）本项目的重要技术成果：也就是在该项目中最重要的技术研究突破，应该对成果进行规范化描述。

（2）主要技术指标的说明：对于项目中各项技术指标，如稳定性、可靠性、安全性、响应时间等方面进行详细的说明与解释，并且主要应说明如何到达。

3．本系统的主要功能描述

该部分主要是对软件系统的主要功能进行一个概要性的描述，其编制时可以参考

《测试分析报告》中“对软件功能的结论”小节的内容。

4．在本项目研发过程中遇到的技术问题和解决方法

这是技术研究报告中最重要的一部分，也是最有价值的一部分。在此应该对每一个技术问题作为一个单独的小节，在每个小节里充分地描述技术问题现象、影响、发生环境等，然后进行充分的分析，尽量将导出解决方法的思路写出来。这样才能够为下次遇到相类似问题提供良好的支持。

5．本系统目前达到的水平、现状、存在的问题及解决方案

上个小节是对已经解决的问题进行描述，而本节则是在前面的基础上进行提升。总结性的评价本系统目前达到的水平，对并该系统的现状进行分析，指出还存在的问题，以及预期的解决方案，并说明为什么现有系统未能有效解决的原因与限制。

6．项目总结

最后，在以上的所有信息的基础上，做出客观、有效、总结性的描述，以便读者能够更好地了解项目的总体情况。

由于《技术报告》的读者的层次包括管理层和技术层两方面，因此他们的专业技能情况不尽相同，因此在技术报告的编制过程中应该注意做到深入浅出，即一方面需要将问题分析透彻，尽量能够挖掘出比较有价值的内容；另一方面还应该在遣词用句时注意通俗化，用更加直观明了的语言把问题解释得清晰易懂。

由于技术报告的特殊性，对于编制人员的要求还是挺高的。应该具有站在较高的层面上对技术问题有宏观性的了解与把握，需要能够有效地采用现实生活中的类比事物进行讲解，以帮助各个层面的读者更好地理解内容。

另外，由于技术报告在许多方面很强调创新性，因此这也对编制人员的专业技术水平提出了很高的要求。可以这么说，技术报告的编制水平反映了一个开发团队的技术研究、技术创新能力。而在系统分析师考试中的论文部分，就是对参考人员在技术报告方面的能力做出了较高的要求。

10.11 开发进度记录

只有计划，没有跟踪与监控，那么项目必将失控。因此，我们需要对资源、经费的使用情况，以及项目的开发进度进行跟踪与监控。因此，一个行之有效的进度监控制度是十分重要的，相应的、合理的、简明的、易于操作的开发进度记录文档也就显得十分重要。

编制《开发进度记录》的主要目的就是及时地向有关管理部门汇报项目开发的进展情况，以便及早地发现、处理问题。而编制的周期可以视项目的规模而定，不过至少是

每月必须形成一个相对比较规范和正式的进度报告。

在 GB 8567—2006 标准中，提供了一个《开发进度记录月报》的文档模板和编写指南，其中规定了以下几个部分内容。

1．标题

包括软件系统的名称、分项目的名称、项目负责人、月报填写人、月报编号、报告年月等基本信息。

2．工程进度与状态

在该部分中应该列出本月内进行的各项主要活动、重要事件、里程碑情况。并且应该用最简明的方法指出项目的状态，即实际的工作进度与原计划相比，是提前、按期还是延误。如果与计划不一致（包括提前、延误），必须说明原因以及准备采用的措施。

3．资源使用与状态

除了进度之外，资源的使用情况也是记录与汇报的重要内容。资源主要包括工时与机时两个方面。工时就是人员使用情况，包括管理用工时、服务用工时、开发用工时三个方面。而机时主要是指设备的使用情况。

类似的，对于资源使用的状态也需要进行定性的描述，就是与原计划相比，是超出、一致，还是节省。如果与计划不一致，那么也需要进行相应的解释以及说明将要采取的措施。

4．经费支出与状态

成本核算是项目监控的重要组成部分。对于项目的经费使用通常包括支持性费用和设备购置费两块，支持性费用包括房租、人员费用（工资、奖金、补贴）、培训费、会务费、招待费、差旅费及其他相关费用；设备购置费用包括软件、新购硬件及原有硬件的折旧。

类似的，对于经费的支出状态也需要进行定性的描述，也就是与原计划相比，是超支，正好，还是节余。如果与计划不一致，也需要进行相应的说明，并提出解决方案。

5．下个月的工作计划

计划必须是动态的调整的，因此在进度报告中还是应该对下个月的工作计划进行一次说明，并及时地反馈到项目总计划中去。

6．问题与建议

最后，还应该将本月中遇到的重要问题、应引起重视的问题进行列举，并提出合理化的建议。

《开发进度记录》是在开发过程中周期性产生的，这个周期可长可短，根据项目的实际情况进行确定。而且开发记录的详略程度也是可以自行根据项目情况进行调整的，主要是在不影响开发工作与及时获得进度状态两个目标之间取平衡点。

通常编制开发进度记录包括两个层面，一个项目经理向管理层/客户提交进度记录，

另一个层面是项目内各个组队向项目经理提交进度记录。对于前一种进度记录应该强调可读性，尽可能以通俗易懂的语言、简单的图表来表示。而对于后一种，应该注重实用性，而不是花哨。

10.12 项目开发总结报告

项目总结是经常容易被遗忘的一环，因为一切都已经成为过去，谁也不愿意去揭起"伤疤"，希望这一切早日过去。因此即使进行项目总结，也是经常是轻描淡写，表表功劳。但项目总结最有价值的东西就是经验、教训的总结，这样可以使得团队在今后的项目中避免相同的问题的出现。

而编制《项目开发总结报告》就是希望将这些宝贵的东西文档化存储下来，以便总结经验、提升团队能力。当然也是对软件开发结果进行评价的一些参考材料。

在 GB 8567—2006 标准中，提供了一个《项目开发总结报告》的文档模板和编写指南，其中规定在项目开发总结报告中除了引言部分之外，应该包括以下几个部分内容。

1．实际开发结果

本小节重点是对已经完成的工作进行一个概要性的总结，通常可以包括以下部分。

（1）产品：包括程序的名字、版本、文件名称、数据库等。

（2）主要功能和性能：列出本软件产品所实际具有的主要功能，以及相关的性能指标。可以对照可行性研究报告、项目开发计划、需求规格说明书和相关内容进行描述，定性地说明开发目标是否达到，或者是未完全达到。

（3）基本流程：用图表的形式说明系统最终的基本处理流程。

（4）进度：定性地说明实际的开发进度执行情况。

（5）费用：定性地说明实际的费用开支执行情况。

2．开发工作评价

本部分将针对开发工作的情况进行一个简要性的评价，主要包括以下几个方面。

（1）对生产效率的评价：包括 KLOC/人月或 FP/人月。

（2）对产品质量的评价：说明在测试中检查出来的错误发生率（Bug/KLOC），另外还应与质量保证计划进行结合说明。

（3）对技术方法使用情况的评价：针对开发中所使用的技术、方法、工具、手段的应用情况进行综合评价。

（4）出错原因分析：对开发中出现的错误的原因进行分析。

3．经验与教训

这是项目开发总结报告中最有价值的一部分，对本次开发中获得主要经验与教训进

行整理，以便对下次开发工作提供指导。

显然，编制《项目开发总结报告》一定是在项目开发完成之后，不过应该在项目开发完成后不久就组织项目总结会，然后形成文档。编写该报告的主要负责人应该是项目经理，其素材应该来源于总结会。

本章参考文献

[1] 张友生. 系统架构设计师教程. 北京：电子工业出版社，2006

第 11 章　项 目 管 理

项目管理是近几十年发展起来的一个管理学分支，同时也变成了一个新兴的行业。它不但应用于国民经济的各部门、社会活动的各方面，而且还与人们的日常生活紧密相连。系统分析师是 IT 项目中的骨干，所以考试大纲中对项目管理知识的要求也越来越多。

在当今社会中，一切都是项目，一切也将成为项目。不管是日常工作还是茶余饭后，人们谈论最多的事情也是各种各样的项目。

11.1　项目与项目管理

项目是在特定条件下，具有特定目标的一次性任务，是在一定时间内，满足一系列特定目标的多项相关工作的总称。项目的定义包含三层含义：第一，项目是一项有待完成的任务，且有特定的环境与要求；第二，在一定的组织机构内，利用有限资源（人力、物力、财力等）在规定的时间内完成任务；第三，任务要满足一定性能、质量、数量、技术指标等要求。

11.1.1　项目概述

根据项目的定义，项目的目标应该包括成果性目标和约束性目标。成果性目标都是由一系列技术指标来定义的，例如，性能、质量、数量、技术指标等；而项目的约束性目标往往是多重的，例如，时间、费用等。项目的目标就是满足客户、管理层和供应商在时间、费用和性能上的不同要求。因此，项目的总目标可以表示为一个空间向量。

1．项目与运作

每个项目实际都是有待完成的任务。事实上，现实生活中，任务一般有两种类型，一类是连续不断、周而复始的重复性活动，人们称之为“运作”（operations），例如，企业日常的生产产品的活动；另一类才是所谓的“项目”（projects），是一次性的活动。表 11-1 列出了项目和运作的不同。

表 11-1　项目与运作的比较

项　　目	运　　作
独一无二	重复的
有限时间	无限时间（相对）
革命性的改变	渐进性的改变
不均衡	均衡

续表

项　　目	运　　作
目标之间不均衡	均衡
多变的资源需求	稳定的资源需求
柔性的组织	稳定的组织
效果性	效率性
以完成目标、目的为宗旨	以完成任务、指标为宗旨
风险和不确定型	经验性

2．项目的属性

不难看出，作为在特定的环境与限制下，有待完成的一次性任务，项目具有如下基本的属性：

（1）一次性。一次性是项目与其他重复性的操作、运行工作的最大区别。项目大多带有某种创新的性质，有明确的起点和终点，过去没有完全可以照搬的先例，将来也不会再有完全相同的重复。项目的其他属性也是从一次性这一主要属性中衍生出来的。

（2）独特性。项目的独特性可能表现在项目的目标、环境、条件、组织、过程等诸多方面。每个项目都有其特别的地方，没有两个项目会是完全相同的。即使有些项目所提供的产品和服务是类似的，但项目的目标、环境、条件、组织、过程等不会完全相同。

（3）目标的确定性。项目必定有明确的目标，没有明确的目标，行动就没有方向，也就不成其为一项任务，也就不会有项目的存在。项目目标一般由成果性目标与约束性目标组成。其中，成果性目标是项目的来源，也是项目的最终目标，在项目实施过程中成果性目标被分解成为项目的功能性要求，是项目全过程的主导目标；约束性目标通常又称限制条件，是实现成果性目标的客观条件和人为约束的统称，是项目实施过程中必须遵循的条件，从而成为项目实施过程中管理的主要目标。

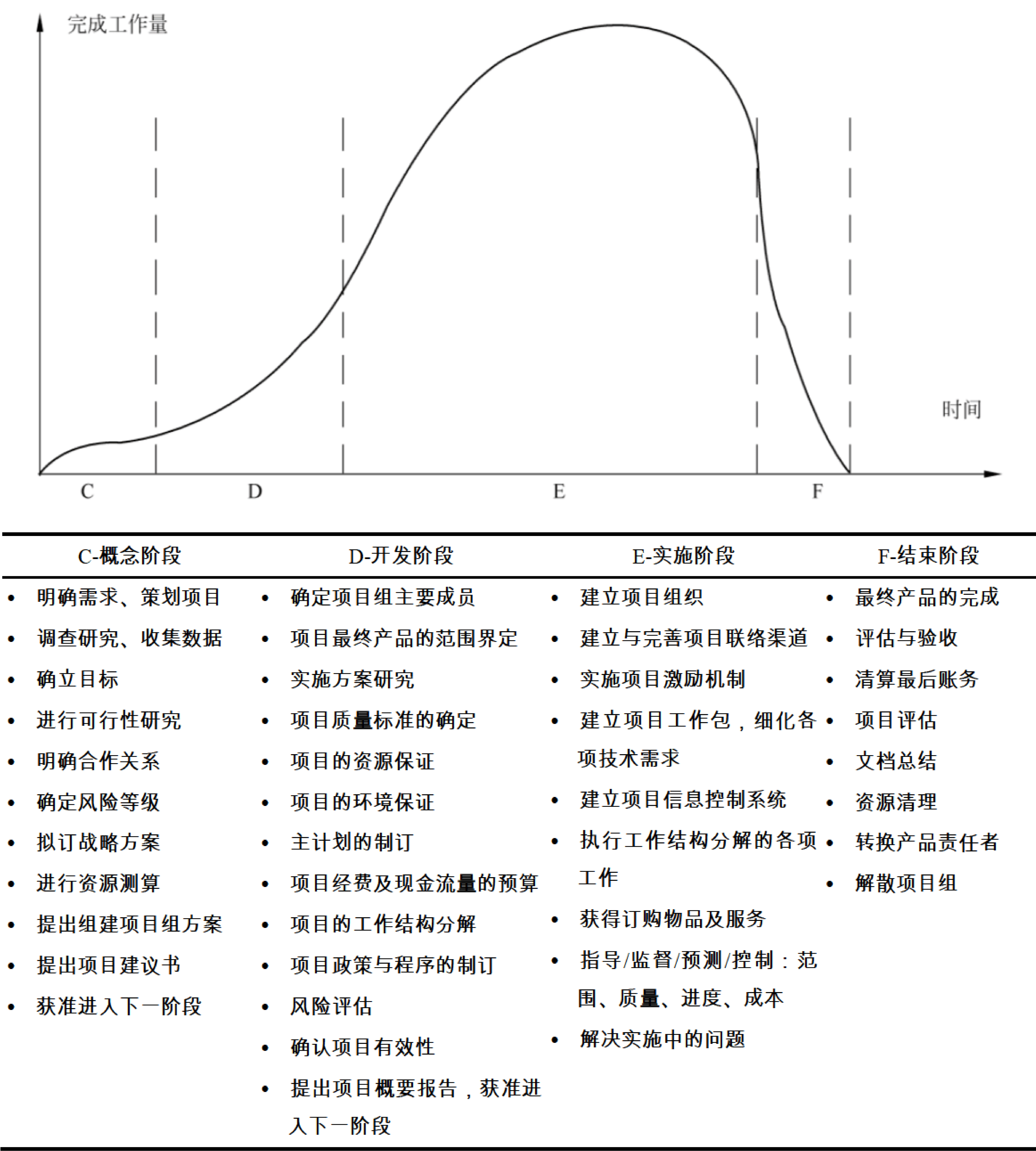
（4）组织的临时性和开放性。因为项目是一次性的，所以项目团队一般也是临时性的。项目执行过程中团队的人数、成员和职能在不断的变化，甚至某些项目团队的成员是借调来的，项目结束时项目团队要解散，人员要转移。项目组织是开放性的，没有严格的边界。参与项目的组织往往有多个，几十个，甚至几百个。它们通过合同、协议以及其他的社会联系组合在一起。这一点和一般的企、事业单位组织很不一样。

（5）成果的不可挽回性。项目不像其他事情可以试做，做坏了可以重来；也不像批量产品，合格率 99.99%是很好了。项目必须确保成功。这是因为在项目的特定条件下，个人和组织的资源有限，一旦失败就永远失去了重新实施原项目的机会。因此，项目具有较大的不确定性，它的过程是渐进的，潜伏着各种风险，要有精心的设计、精心的制作和精心的控制，才能达到预期的目标。

3．项目的生命周期

项目的生命周期由项目阶段构成，不同的项目可能包含不同的阶段。一般来说项目

的生命周期有几个基本的阶段：概念阶段（Conception Phase）、开发阶段（Development Phase）、实施阶段（Execute Phase）及结束阶段（Finish Phase）。项目在不同阶段，其管理的内容也不相同。图 11-1 就是从项目生命周期的角度，对项目的 C，D，E，F 4 个阶段工作内容的概括描述。



C-概念阶段	D-开发阶段	E-实施阶段	F-结束阶段
• 明确需求、策划项目	• 确定项目组主要成员	• 建立项目组织	• 最终产品的完成
• 调查研究、收集数据	• 项目最终产品的范围界定	• 建立与完善项目联络渠道	• 评估与验收
• 确立目标	• 实施方案研究	• 实施项目激励机制	• 清算最后账务
• 进行可行性研究	• 项目质量标准的确定	• 建立项目工作包，细化各项技术需求	• 项目评估
• 明确合作关系	• 项目的资源保证	• 建立项目信息控制系统	• 文档总结
• 确定风险等级	• 项目的环境保证	• 执行工作结构分解的各项工作	• 资源清理
• 拟订战略方案	• 主计划的制订	• 获得订购物品及服务	• 转换产品责任者
• 进行资源测算	• 项目经费及现金流量的预算	• 指导/监督/预测/控制：范围、质量、进度、成本	• 解散项目组
• 提出组建项目组方案	• 项目的工作结构分解	• 解决实施中的问题	
• 提出项目建议书	• 项目政策与程序的制订		
• 获准进入下一阶段	• 风险评估		
	• 确认项目有效性		
	• 提出项目概要报告，获准进入下一阶段		

图 11-1　项目的生命周期及其主要工作

11.1.2　项目管理概述

项目管理就是把各种资源应用于目标，以实现项目的目标，满足各方面既定的需求。项目管理首先是管理，所以管理学的一般理论照样适合项目管理，不同的是项目管理的

管理对象是项目；管理的方式是目标管理；项目的组织通常是临时性、柔性、扁平化的组织；管理过程贯穿着系统工程的思想；管理的方法工具和手段具有先进性和开放性，用到多学科的知识和工具。

项目管理的要素有环境、资源、目标和组织。

（1）环境。首先，项目不是空中楼阁，都是在特定的环境下进行的。项目管理者必须对项目所处的外部环境有正确的认识。项目的外部环境包括自然、技术、政治、社会、经济、文化以及法律法规和行业标准等。

（2）资源。资源的概念内容十分丰富，可以理解为一切具有现实和潜在价值的东西，包括自然资源和人造资源、内部资源和外部资源、有形资源和无形资源。诸如人力、材料、设备、资金、信息、信息技术、市场等。

（3）目标。如前所述，项目的目标就是满足客户、管理层和供应商等项目干系人在时间、费用和性能上的不同要求。

（4）组织。组织就是把多个人联系起来，做一个人无法做的事，是管理的一项功能，组织包括与它要做的事相关的人和资源，及其相互关系。项目组织与其他组织一样，要有好的领导、章程、沟通、人员配备、激励机制，以及好的组织文化等。同时，项目组织也有其与其他组织不同的特点。

与传统的部门管理相比，项目管理的最大特点是注重于综合性管理，并且项目管理工作有严格的时间期限。具体来讲，表现在以下几个方面：

（1）项目管理的对象是项目或被当作项目来处理的运作。项目管理是针对项目的特点而形成的一种管理方式，因此其适用对象是项目，特别是大型的、比较复杂的项目；鉴于项目管理的科学性和高效性，有时人们会将重复性的“运作”中某些过程分离出来，加上起点和终点当作项目来处理，以便于在其中应用项目管理的方法。

（2）项目管理的全过程都贯穿着系统工程的思想。项目管理把项目看成一个完整的系统，依据系统论“整体-分解-综合”的原理，可将系统分解为许多责任单元，由责任者分别按要求完成目标，然后汇总、综合成最终的成果；同时，项目管理把项目看成一个有完整生命周期的过程，强调部分对整体的重要性，促使管理者不要忽视其中的任何阶段以免造成总体的效果不佳甚至失败。

（3）项目管理的组织具有特殊性。项目管理的一个最为明显的特征是其组织的特殊性，项目管理的组织是临时性的、开放的。项目管理的组织结构多为矩阵结构，而非直线职能结构。

（4）项目管理的体制是一种基于团队管理的个人负责制。由于项目系统管理的要求，需要集中权力以控制工作正常进行，因此项目经理是一个关键角色。

（5）项目管理的方式是目标管理。项目管理是一种多层次的目标管理方式。由于项目往往涉及的专业领域十分宽广，而项目管理者谁也无法成为每一个专业领域的专家，对某些专业虽然有所了解但不可能像专门研究者那样深刻。项目管理者只能以综合协调

者的身份，向被授权的专家，讲明应承担工作责任的意义，协商确定目标以及时间、经费、工作标准的限定条件。此外的具体工作则由被授权者独立处理。

（6）项目管理的要点是创造和保持一种使项目顺利进行的环境。项目管理是一个管理过程，而不是技术过程，处理各种冲突和意外事件是项目管理的主要工作。所以，有人认为，项目管理就是创造和保持一种环境，使置身于其中的人们能在集体中一道工作以完成预定的使命和目标。

（7）项目管理的方法、工具和手段具有先进性、开放性。项目管理采用科学先进的管理理论和方法。如采用网络图编项目进度计划，采用目标管理、全面质量管理、价值工程、技术经济分析等理论和方法控制项目总目标；采用先进高效的管理手段和工具，主要是使用电子计算机进行项目信息处理等。

11.2 项目范围管理

从某种意义上讲，目标不明确、范围不清晰的项目永远都不可能实施成功。所以，项目一开始就应该把目标和范围定清楚，否则就不宜开始实施。范围不清晰，实际上等同于项目没有成功标准，因为范围决定项目成本和时间，范围变化，成本和进度必然跟着变化，而这三者又构成项目成功的三要素，所以，范围不清晰的项目本质上不会成功。按说这是一个相当明显的逻辑，但遗憾的是，范围不清晰的问题在 IT 项目中比较突出，因为 IT 项目是无形的，很多时候客户的需求是朦胧的，客户不具备把他们的需求表达得很具体的能力，他甚至说不出来具体需要哪些功能模块，这成为信息化项目最大的特点和难点。

所以，在项目一开始要先进行范围计划。范围计划的目的是与最终用户及项目出资人明确确认项目目标和主要子项目，使项目团队与客户之间达成一个协议。范围计划的主要工作就是确定项目范围并编写范围说明书。项目范围说明书说明了为什么要进行这个项目，明确了项目的目标和主要的可交付成果，是将来项目实施的重要基础。一般来说，项目范围说明书要由项目团队来写。项目范围说明书是项目团队和任务委托者之间签订协议的基础。

11.2.1 项目范围计划

一般而言，编写项目范围说明书时必须了解以下情况，作为范围计划的依据：

（1）技术规格说明书。技术规格说明书一般和项目合同据有同等的法律效力，是任务的委托者在项目结束，或者项目阶段结束时要求项目团队交出的成果。

（2）项目许可证书。项目许可证中有关于项目目标的要求，即当初项目论证时的最根本要求。

（3）有委托签约的项目，合同条款及交付要求。

（4）制约因素和假设前提。在编写范围说明书时，项目团队需要考虑有哪些因素限制自己的行动。

1．工具和技术

进行范围计划，可以使用的工具和技术有以下一些：

（1）成果分析。通过成果分析可以加深对项目成果的理解，确定其是否必需、是否多余以及是否有价值，可以采用价值点分析、比较分析和质量功能展示等技术和手段。

（2）成本效益分析。估算项目的有形成本、无形成本与收益，然后，用投资收益率或投资回收期等经济方法，评估这些经确认的选择方案的相对优势。

（3）项目方案识别技术。项目方案识别技术泛指提出实现项目目标的方案的所有技术，可以采用头脑风暴法，并从中比较甄选出最合适的技术。

（4）领域专家。可以请领域专家对各种方案进行评价。任何经过专门训练或具有专门知识的集体或个人均可视为领域专家。

2．主要成果

范围计划输出的主要成果有两个，分别是范围说明书和范围管理计划。

范围说明书为将来项目实施提供了基础。随着项目的进展，需要对范围说明书进行修改和细化，以反映项目和外部环境的变化。范围说明书的内容应当包括：

（1）项目合理性说明。解释为何要进行这一项目，为以后权衡各种利弊关系提供依据。

（2）可交付成果清单。列入清单中的事项一旦圆满地完成，并交付给使用者——中间用户或最终用户，就标志着项目阶段或项目的完成。

（3）项目目标。即一个公认的项目验收标准，可以量化衡量的目标。至少要让项目干系人看到，原定的费用、进度和质量目标均已达到。

范围管理没有小事情，项目最终的成果是逐步形成的，在这个过程中，变化既是好事也是坏事，好处是毕竟项目在进行中，过程中及时变化总比结果成型后变化代价小，坏处是变化往往会牵一发动全身，一点变化影响工期和成本。所以项目变化要相当慎重，而范围管理计划的重点就是实现对这些范围变化制定规约，以指导执行。

范围管理计划是项目计划书的一部分。范围管理计划说明如何管理项目范围以及如何将变更纳入到项目的范围之内。范围管理计划还要对项目范围的稳定性进行评价，即项目范围变化的可能性、频率和幅度，并说明如何识别范围变更以及如何将其分类。

11.2.2　工作分解结构

为了便于制定项目各具体领域和整体计划，需要将项目及其主要可交付成果划分成一些较小，更易管理的部分，叫做项目分解。项目分解是在确定了项目的范围之后进行的，项目范围说明书是项目分解的直接依据。项目其他具体领域的计划结果可作为间接依据。例如，如果在做风险管理计划时发现有必要采取风险回避行动，那么，就应当明

确这种行动的主要可交付成果是什么。

1. 主要目的和用途

工作分解结构（Work Breakdown Structure，WBS）的主要目的和用途如下。

（1）明确和准确说明项目的范围。

（2）为各独立单元分派人员，规定这些人员的相应职责。

（3）针对各独立单元，进行时间、费用和资源需要量的估算，提高费用、时间和资源估算的准确性。

（4）为计划、预算、进度安排和费用控制奠定共同基础，确定项目进度测量和控制的基准。

（5）将项目工作与项目的财务账目联系起来。

（6）便于划分和分派责任，自上而下将项目目标落实到具体的工作上，并将这些工作交给项目内外的个人或组织去完成。

（7）确定工作内容和工作顺序。

（8）估算项目整体和全过程的费用。

项目工作分解是一项特别专业的工作，进行过程中需要有足够的专业知识和项目经验，好的工作分解能提高效率，不好的工作分解则增加执行的摩擦，降低效率，为执行阶段设下很多埋伏。

2. 工作步骤

一般而言，项目工作分解有如下几个步骤。

（1）识别项目主要组成部分，包括项目的可交付成果和项目管理本身。回答问题：要实现项目的目标需要完成哪些主要工作？主要工作指贯彻项目生命期始终的大块工作，在 WBS 中列在第二层。

（2）如果各层次上的工作经过划分，能够估算出完成它们各组成部分所需的费用和时间，则进行下面的第（4）步，否则进行第（3）步。

（3）找出上述各组成部分更小的组成部分。回答问题：要完成上述各组成部分，有哪些更具体的工作需要做。对于各组成部分的更小构成部分，应该说明需要取得哪些切实的、可以核查的结果以及完成这些更小构成部分的先后顺序。对于这些更小的构成部分，重复第（2）步。

（4）检查划分后得到的各更小组成部分，需检查：

- 不进行这一层次的工作，上一层次的各项工作能否完成？完成了这些工作，上一层次的各项工作就一定能完成吗？如果不进行这一层次的工作，上一层次的各项工作也能完成，或者即使完成了这一层次的所有工作，上一层次的工作还是不能完成，则必须对上一层次各项工作的划分进行修改，重新划分、增加或者删除一些事项。
- 这一层次各项工作的内容、范围和性质是否都已明确，是否已写出了简要的范围

说明书？若不是，则必须弄清楚，必须对相应的范围说明书进行修改和补充。

项目的工作结构分解，特别是较大项目，应该注意如下几点：

（1）确定 WBS 的过程就是将项目的产品或服务、组织和过程这三种不同结构综合为项目分解结构的过程。项目经理和团队成员要善于巧妙地将项目按其产品或服务的结构划分、按项目的阶段划分以及按项目组织的责任划分有机地结合起来。

（2）最底层的工作块应当便于完整无缺地分派给项目内外的不同个人或组织，所以，要求明确各工作块之间的界面。界面清楚有利于减少项目进展过程中的协调工作量。

（3）最底层的工作块应当非常具体，以便各工作块的承担者都能明确自己的任务、努力的目标和承担的责任。工作块划分得具体，也便于监督和业绩考核。

（4）逐层分解项目或其主要可交付成果的过程实际上也是分派角色和职责的过程。

希赛教育专家提示：项目分解完成之后，必须交出的成果就是 WBS。WBS 中的每一项工作（或者称为单元）都要编上号码。

11.2.3　项目范围确认和控制

范围确认是通过参与者（倡议者、委托人和顾客等）的行为正式确定项目范围的过程。它要求回顾生产工作和生产成果，以保证所有项目都能准确地、满意地完成。如果这个项目已提前终止，这个范围确认过程也应该证实并应以书面文件的形式把它的完成情况记录下来。

一个典型的软件实施项目，其范围可以通过以下几个维度来确定。

（1）组织范围：也就是将来软件系统所覆盖的所有部门或二级机构，部门要具体到科室和岗位，二级机构要指明地点、科室和岗位。

（2）功能范围：就是本期软件系统实施都使用哪些功能模块，以及这些功能模块的许可数（站点数）。

（3）业务流程范围：描述客户方需要通过系统实现的日常业务处理，覆盖系统可以实现的客户业务。需要有一个客户组织中所有部门和二级单位所使用的功能模块对照表，用于描述实施单位各自启用的系统模块。由于集团型企业各独立核算单位的业务类型、管理重点可能各不相同。因此，购买的软件模块并非集团内所有单位都要应用。

（4）接口范围：描述本系统和客户方已有的其他系统的接口集合。接口包括数据库数据共享层面以及系统功能整合层面的接口。

（5）数据导入范围：将来软件系统要从客户已有的一些管理工具或系统中导入哪些数据，以及数据导入的方法等。

（6）客户化补充开发范围：需要界定客户个性化开发的需求范围。

（7）硬件及网络系统部署范围：客户方的硬件和网络系统如果需要项目组完成，那么应该明确范围。

（8）培训范围：需要培训的人员、内容等。

（9）每期的范围：复杂一点的软件项目，需要划分几个阶段来做，所以，有必要明确每个阶段的工作：为当期实现哪些功能。

造成范围变更的原因有很多，主要有以下一些：

（1）项目外部环境发生变化，例如，政府颁布了新法令，竞争对手生产出了新产品，本国货币贬值等。

（2）项目范围计划不周，有错误或遗漏，例如，在设计电话系统时未考虑到计算机网络的使用等。

（3）世界上出现了或设计人员提出了新技术、手段或方案，例如，项目实施后出现了制定范围管理计划时尚未出现的，可大幅度降低成本的新技术。

（4）项目实施组织本身发生变化，例如，项目所在单位同其他单位合并，项目团队人事发生变化等。

（5）项目业主对项目、项目的产品的要求发生变化，例如，业主希望系统灵活定制报表以适应上级单位多变的要求等。

对范围变更进行控制时，要以 WBS、项目进展报告、来自项目内外的变更请求和范围管理计划为依据。进行范围变更控制必须经过范围变更控制系统。

范围变更出现后，应修改有关技术文件和项目计划，并通知有关的项目干系人。对范围变更采取措施，进行处理之后，应当将造成范围变更的原因、采取的措施，以及采取此措施的理由、从此次变更中吸取的教训等都记录在案，形成书面文件，存入本项目和其他项目的数据库。

范围变更控制必须同整体、进度、费用和合同变更控制等其他控制过程紧密结合起来。

11.3 项目时间管理

时间是项目目标三要素之一，严格的时间期限是项目的主要特点之一，因此，时间管理在项目管理中至关重要。从大的方面讲，项目时间管理又包括两部分：进度计划和进度控制，进度计划的目的是为了控制时间和节约时间，通常通过活动定义、活动排序、活动时间估算和进度编排 4 个步骤完成的；进度控制则是在实际执行过程中，检测实际进度和计划进度的差距，并采取措施加以控制。

11.3.1 进度计划编制

在编制进度计划之前，需要先对活动进行定义和排序，对相关资源进行估算。

1．活动定义

为了便于制定项目各具体领域和整体计划，需要将项目及其主要可交付成果分解成一些较小，更易管理和单独完成的部分。项目分解是编制进度计划，进行进度控制的基

础。项目分解就是根据项目状况，采用 WBS 技术，将一个总体项目分解为若干项工作或活动，直到具体明确为止。项目分解是项目管理的一项最基本的工作。项目分解需要足够的专业知识和项目管理经验，一般说来，项目分解应根据项目的具体情况以及进度计划的类型和作用确定。

活动就是 WBS 中确定的工作任务或工作元素。活动界定则是明确实现项目目标需要进行的各项活动。对于一个较小的项目，活动可能会界定到每一个人；但对于一个较大的项目、复杂的项目，如果运用 WBS 技术对项目进行了分解，项目经理就没有必要把每一个具体的活动都界定到每一个人，因为这样会浪费许多时间，甚至会遗漏很多的活动。因此，对于运用 WBS 分解的项目，个人活动可以由工作任务的负责人或责任小组来界定。

在项目分解的基础上，为了更明确地描述项目所包含的各项工作的具体内容和要求，需要对工作进行描述。工作描述作为编制项目计划的依据，同时便于项目实施过程中更清晰地领会各项工作的内容。工作描述的依据是项目描述和 WBS，其结果是工作描述表及项目工作列表。

希赛教育专家提示：为了明确各部门或个人在项目中的责任，便于项目管理部门在项目实施过程中的管理协调，应根据 WBS 和项目组织结构图表对项目的每一项工作或任务分配责任者和落实责任。工作责任分配的结果是形成工作责任分配表。

2．活动排序

一个项目有若干项工作和活动，这些工作和活动在时间上的先后顺序称之为逻辑关系。逻辑关系可分为两类，其一为客观存在的、不变的逻辑关系，也称之为强制性逻辑关系，例如，建一座厂房，首先应进行基础施工，然后才能进行主体施工；其二为可变的逻辑关系，也称为组织关系，这类逻辑关系随着人为约束条件的变化而变化，随着实施方案、人员调配、资源供应条件的变化而变化，例如，一项任务有三项工作 A，B，C，假使 A，B，C 之间不存在不变的逻辑关系，则要完成这一任务，这三者之间的关系有多种不同的方案，显然，不同的统筹安排方案所花费工期、费用各不相同。

所有活动必须被正确地加以排序，以便今后制订可行的进度计划。排序可借助应用软件，也可用手工排序。对于小型项目手工排序很方便，而大型项目可以用手工编制和软件排序相结合的方法完成。

3．活动时间估算

根据项目分解情况，计算各工作或活动的工程量或工作量，包括工作的内容、工作开展的前提条件、工作量、所需的资源等。

工作持续时间是指在一定的条件下，直接完成该工作所需时间与必要停歇时间之和，单位可为日、周、旬、月等。工作持续时间是计算其他网络参数和确定项目工期的基础。工作持续时间的估算是编制项目进度计划的一项重要的基础工作，要求客观正确。如果工作时间估算太短，则会造成被动紧张的局面；相反，则会延长工期。在估算工作

时间时，不应受到工作的重要性及项目完成期限的限制，要在考虑各种资源供应、技术、工艺、现场条件、工作量、工作效率、劳动定额等因素的情况下，将工作置于独立的正常状态下进行估算。

4．进度编排

用网络图的绘制主要是依据项目工作关系表，通过网络图的形式将项目的工作关系表达出来。

在完成了项目分解、确定各项工作和活动先后顺序、计算工程量或工作量并估算出各项工作持续时间的基础上，即可安排项目的时间进度。

11.3.2 计划编制的方法和工具

计划编制的方法和工具主要有关键路径法、计划评审技术、甘特图等。

1．关键路径法

关键路径法（Critical Path Method，CPM）是借助网络图和各活动所需时间（估算值），计算每一活动的最早或最迟开始和结束时间。CPM 法的关键是计算总时差，这样可决定哪一活动有最小时间弹性。CPM 算法也在其他类型的数学分析中得到应用。

CPM 算法的核心思想是将 WBS 分解的活动按逻辑关系加以整合，统筹计算出整个项目的工期和关键路径。

项目活动间存在以下 4 种依赖关系。

（1）结束对起始（FS）：前一活动必须在后一活动开始前结束。

（2）结束对结束（FF）：前一活动必须在后一活动结束前结束。

（3）起始对起始（FS）：前一活动必须在后一活动开始前开始。

（4）起始对结束（FS）：前一活动必须在后一活动结束前开始。

每个活动有以下 4 个和时间相关的参数。

（1）最早开始时间（ES）：某项活动能够开始的最早时间。

（2）最早结束时间（EF）：某项活动能够完成的最早时间。

EF＝ES＋工期估算

（3）最迟结束时间（LF）：为了使项目按时完成，某项工作必须完成的最迟时间。

（4）最迟开始时间（LS）：为了使项目按时完成，某项工作必须开始的最迟时间。

LS＝LF－工期估算

CPM 方法有以下两个规则。

规则 1：某项活动的最早开始时间必须相同或晚于直接指向这项活动的最早结束时间中的最晚时间。

规则 2：某项活动的最迟结束时间必须相同或早于该活动直接指向的所有活动最迟开始时间的最早时间。

根据以上规则，可以计算出工程的最早完工时间。通过正向计算（从第一个活动到最后一个活动）推算出最早完工时间，步骤如下：

（1）从网络图始端向终端计算。

（2）第一任务的开始为项目开始。

（3）任务完成时间为开始时间加持续时间。

（4）后续任务的开始时间根据前置任务的时间和搭接时间而定。

（5）多个前置任务存在时，根据最迟任务时间来定。

通过反向计算（从最后一个活动到第一个活动）推算出最晚完工时间，步骤如下：

（1）从网络图终端向始端计算。

（2）最后一个任务的完成时间为项目完成时间。

（3）任务开始时间为完成时间减持续时间。

（4）前置任务的完成时间根据后续任务的时间和搭接时间而定。

（5）多个后续任务存在时，根据最早任务时间来定。

最早开始时间和最晚开始时间相等的活动成为关键活动，关键活动串连起来的路径成为关键路径，关键路径的长度即为项目的工期。

2．计划评审技术

计划评审技术（Program Evaluation an Review Technique，PERT）的理论基础是假设项目持续时间以及整个项目完成时间是随机的，且服从某种概率分布。PERT 可以估算整个项目在某段时间内完成的概率。由于 PERT 和 CPM 在项目的进度计划中应用非常广，因此，下面将通过一个项目实例对此技术加以介绍。

PERT 对各个项目活动的完成时间按以下三种不同情况估算。

（1）乐观时间（Optimistic Time）。任何事情都顺利的情况下，完成某项工作的时间。

（2）最可能时间（Most Likely Time）。正常情况下，完成某项工作的时间。

（3）悲观时间（Pessimistic Time）。最不利的情况下，完成某项工作的时间。

PERT 认为以上三个估算值服从 β 分布，因此可算出每个活动的期望 t_i：

$$t_i = \frac{a_i + 4m_i + b_i}{6}$$

其中：a_i 表示第 i 项活动的乐观时间，m_i 表示第 i 项活动的最可能时间，b_i 表示第 i 项活动的悲观时间。

根据 β 分布的方差计算方法，第 i 项活动的持续时间方差为：

$$\sigma_i^2 = \frac{(b_i - a_i)^2}{36}$$

例如，希赛在线教育平台系统的建设可分解为需求分析、设计编码、测试、安装部署等 4 个活动，各个活动顺次进行，没有时间上的重叠，活动的完成时间估算如图 11-2 所示。

图 11-2 中每个箭头下给出的 3 个数字分别代表 a_i、m_i 和 b_i 的数值。则各活动的期望工期和方差为：

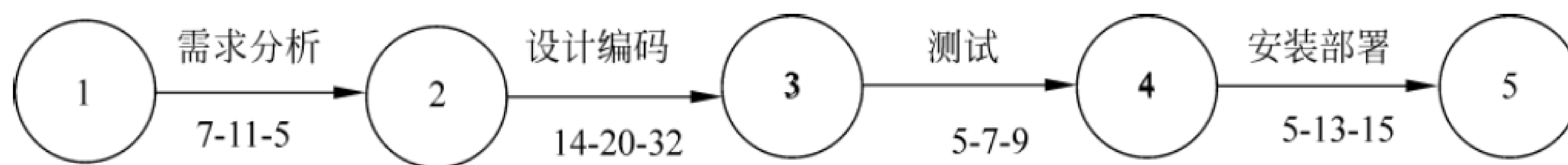


图 11-2 工作分解和活动工期估算

$$t_{需求分析}=\frac{7+4\times11+15}{6}=11，\quad \sigma^2_{需求分析}=\frac{(15-7)^2}{36}=1.778$$

$$t_{设计编码}=\frac{14+4\times20+32}{6}=21，\quad \sigma^2_{设计编码}=\frac{(32-14)^2}{36}=9$$

$$t_{测试}=\frac{5+4\times7+9}{6}=7，\quad \sigma^2_{测试}=\frac{(9-5)^2}{36}=0.445$$

$$t_{安装部署}=\frac{5+4\times13+15}{6}=12，\quad \sigma^2_{安装部署}=\left(\frac{15-5}{6}\right)^2=2.778$$

认为整个项目的完成时间是各个活动完成时间之和，且服从正态分布。整个项目完成的时间 t 的数学期望 T 和方差 σ^2 分别等于：

$$\sigma^2=\sum\sigma_i^2=1.778+9+0.445+2.778=14.001$$

$$T=\sum t_i=11+21+7+12=51\text{PERT}$$

标准差为：

$$\sigma=\sqrt{\sigma^2}=\sqrt{14.001}=3.742$$

据此，可以得出正态分布曲线，如图 11-3 所示。因为图 11-3 是正态曲线，根据正态分布规律，在±σ 范围内，即在 47.258～54.742 天之间完成的概率为 68.26%；在±2σ 范围内，即在 43.561～58.484 天完成的概率为 95.43%；在±3σ 范围内，即 39.774～62.226 天完成的概率为 99.73%。如果客户要求在 39 天内完成，则可完成的概率几乎为 0，也就是说，项目有不可压缩的最小周期，这是客观规律。

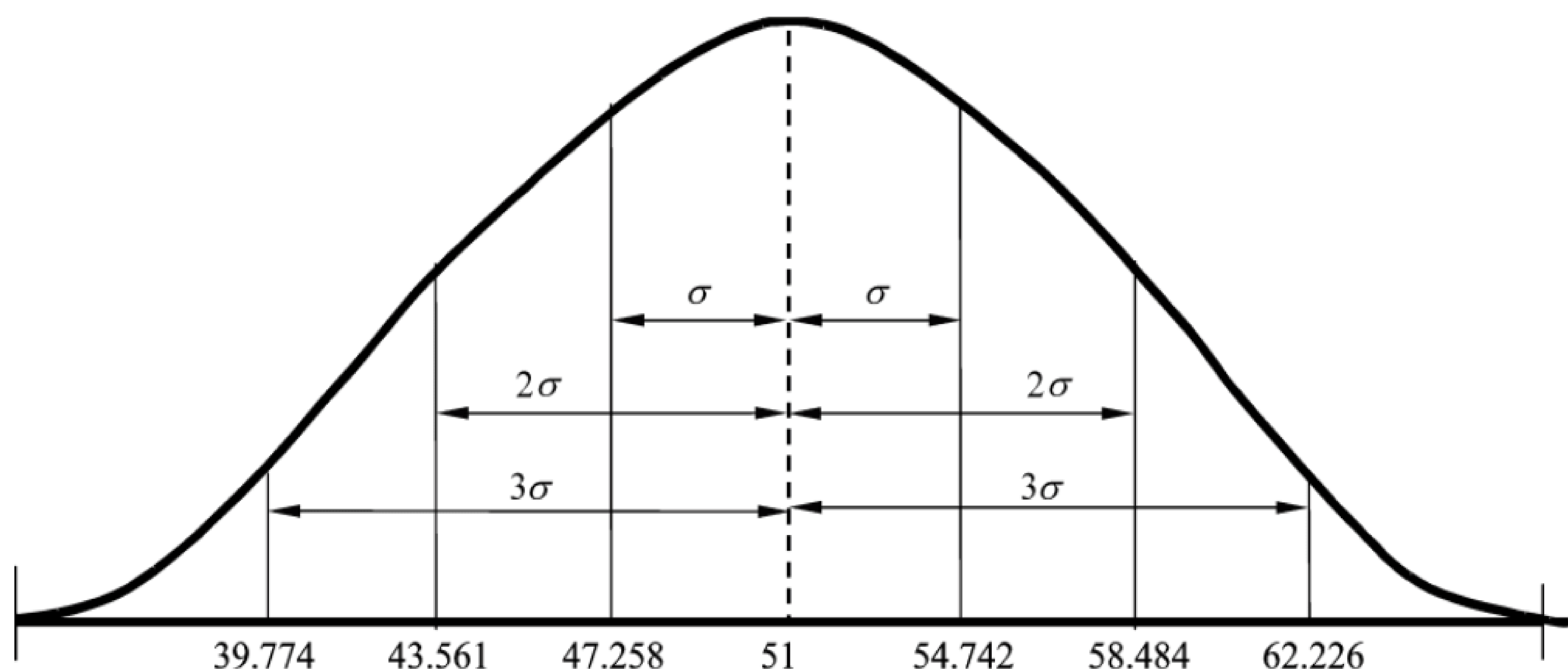


图 11-3 OA 项目的工期正态分布图

通过查标准正态分布表，可得到整个项目在某一段时间内完成的概率。例如，如果客户要求在 60 天内完成，那么可能完成的概率为：

$$P\{t \le 60\} = \phi\left(\frac{60-T}{\sigma}\right) = \phi\left(\frac{60-51}{3.742}\right) = 0.992$$

如果客户要求再提前 7 天，则完成的概率为：

$$P\{t \le 53\} = \phi\left(\frac{53-T}{\sigma}\right) = \phi\left(\frac{53-51}{3.742}\right) = 0.7019$$

实际上，大型项目的工期估算和进度控制非常复杂，往往需要将 CPM 和 PERT 结合使用，用 CPM 求出关键路径，再对关键路径上的各个活动用 PERT 估算完成期望和方差，最后得出项目在某一时间段内完成的概率。PERT 还告诉我们，任何项目都有不可压缩的最小周期，这是客观规律，千万不能不顾客观规律而对用户盲目承诺，否则，必然会受到客观规律的惩罚。

3．甘特图

甘特图也叫做线条图或横道图，它是以横线来表示每项活动的起止时间，如图 11-4 所示。甘特图的优点是简单、明了、直观，易于编制，因此到目前为止仍然是小型项目中常用的工具。即使在大型工程项目中，它也是高级管理层了解全局、基层安排进度时有用的工具。

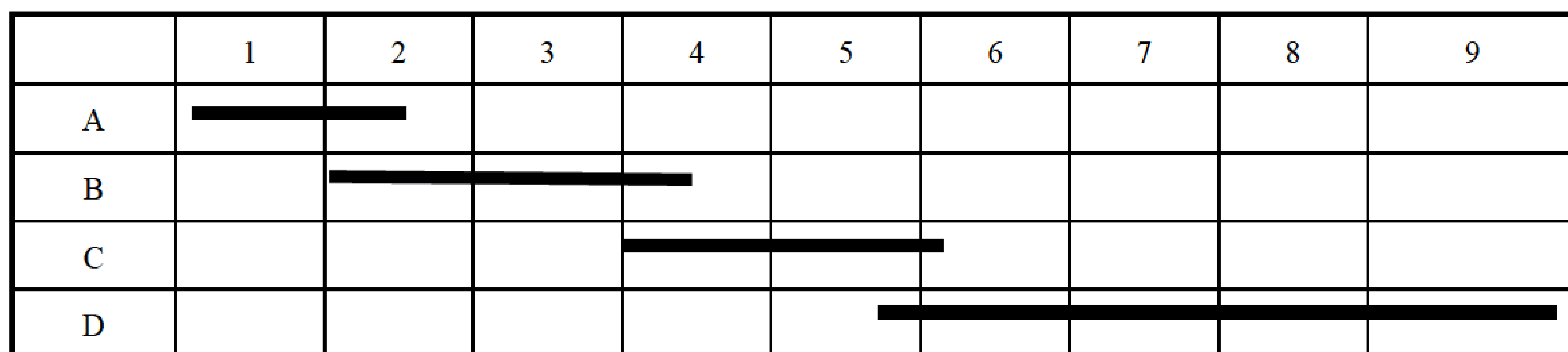


图 11-4　甘特图

在甘特图上，可以看出各项活动的开始和终了时间。在绘制各项活动的起止时间时，也考虑它们的先后顺序。但各项活动上间的关系却没有表示出来，同时也没有指出影响项目寿命周期的关键所在。因此，对于复杂的项目来说，甘特图就显得不足以适应。

11.3.3　项目进度控制

项目进度控制是非常困难的事情，IT 项目的拖期几乎成了普遍现象，尽管项目管理人员做出各种各样的努力，还是不能避免项目拖延的命运。笔者认为，造成项目拖期的主要原因有两大类，一类是初期估算工作量有误，项目难度考虑不足或者估算过于乐观；另一类则是缺乏有效的进度检查和控制手段，下面将简要介绍一下项目进度控制的流程和要领。

1．项目进度控制流程

编制进度计划的目的，就是指导项目的实施，以保证实现项目的工期目标。但在进度计划实施过程中，由于主客观条件的不断变化，计划亦需随之改变。凭借一个最优计划来实现一劳永逸的工作是不可能的。因此，在项目进行过程中，必须不断监控项目的进程以确保每项工作都能按进度计划进行；同时必须不断掌握计划的实施状况，并将实际情况与计划进行对比分析，必要时应采取有效的对策，使项目按预定的进度目标进行，避免工期的拖延。这一过程称之为进度控制。该过程可用图 11-5 加以描述。

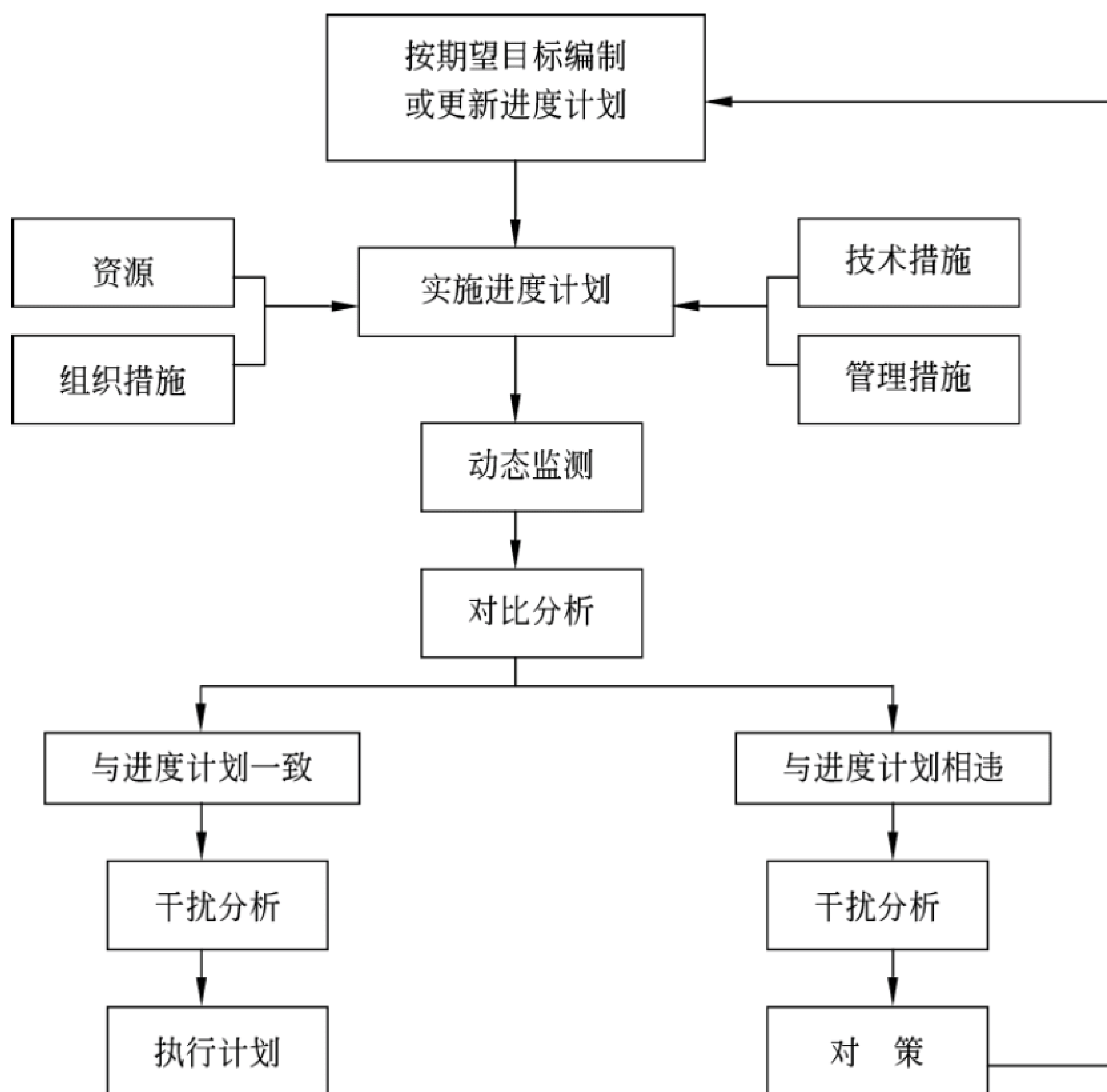


图 11-5 项目进度控制过程

有效进行项目进度控制的关键是监控实际进度，及时、定期地将实际进度与进度计划进行比较，并及时采取纠正措施。项目管理人员不能简单地认为问题会在不采取任何措施的情况下自动消失。项目的进度控制就是在既定工期内，编制出最优的进度计划，在执行计划的过程中，经常检查项目实际进度情况，并将其与进度计划相比较，若出现偏差，便分析产生的原因及对工期的影响程度，确定必要的调整措施，更新原计划。这一过程不断地循环，直至项目完成。项目进度控制的目标就是确保项目按既定工期目标实现，或是在保证项目质量并不因此而增加项目实际成本的条件下，适当缩短项目工期。

项目进度控制的主要方法是计划、控制和协调。计划是指确定项目总进度控制目标和分进度控制目标，并编制其进度计划；控制是指在项目实施全过程，进行的检查、比

较及调整；协调是指协调参与项目的各有关单位、部门和人员之间的关系，使之有利于项目的进展。

进度控制所采取的措施主要有组织措施、技术措施、合同措施、经济措施和管理措施等。组织措施是指落实各层次的进度控制人员，具体任务和工作责任；建立进度控制的组织系统；按照项目的结构、工作流程或合同结构等进行项目的分解，确定其进度目标，建立控制目标体系；确定进度控制工作制度，如检查时间、方法、协调会议时间、参加人员等；对影响进度的因素进行分析和预测。技术措施主要是指采取加快项目进度的技术方法。合同措施是指项目的发包方和承包方之间，总包方与分包方之间等通过签订合同明确工期目标，对项目完成的时间进行制约。经济措施是指实现进度计划的资金保证措施。管理措施是指加强信息管理，不断地收集项目实际进度的有关信息资料，进行整理统计，与进度计划相比较，并定期提出项目进展报告，以此作为决策依据之一。项目计划中的某些东西在付诸实施后才会发现无法实现。即使勉强实现，也要付出很高的代价。遇到这种情况，就必须对项目计划进行修改，或重新计划。在项目实施过程中要进行多次计划（P）、实施（D）、检查（C）和行动（A）循环。

2．项目进度控制要领

项目有一个重要属性就是：一次性的工作，这一属性使得项目控制有别于其他管理控制。企业生产或业务活动可事先制定出明确的标准，在生产或业务活动过程中，管理人员将实施的实际情况同这些标准进行对照，由此发现计划的偏离程度。但是，项目活动由于一般无先例，事先不能制定出明确的标准。所以，项目常常根据投入的多少，例如费用、人力或其他资源来评价实际实施结果，通过协商和判断来进行控制。

在项目实施过程中，必须定期对项目的进展进行测量，找出偏离计划之处，将其反馈到有关的控制子过程中。项目计划中的某些东西在付诸实施后才会发现无法实现。即使勉强实现，也要付出很高的代价。遇到这种情况，就必须对项目计划进行修改，或重新计划。项目控制要真正有效，就必须做到：

（1）要有明确的目的。项目控制的基本目的就是保证项目目标的实现，实现项目的范围、进度、质量、费用、风险、人力资源、沟通、合同等方面的目标。

（2）要及时。必须及时发现偏差，迅速报告项目有关方面，使他们能及时做出决策，采取措施加以更正。否则，就会延误时机，造成难以弥补的损失。

（3）要考虑代价。对偏差采取措施，甚至对项目过程进行监督，都是需要费用的。因此，一定要比较控制活动的费用和可能产生的效果。只有在收效大于费用时才值得进行控制。

（4）要适合项目实施组织和项目团队的特点。控制要同人员分工、职责、权限结合起来。要考虑控制的程序、做法、手段和工具是否适合项目实施组织和项目团队成员个人的特点，是否能被他们接受。控制要对项目各项工作进行检查，要采取措施进行纠正等。所有这些都要涉及到人。人们是不愿意接受使他们不愉快的控制措施的。实施控制

的项目经理或其他成员应当懂点心理学，弄清他们为什么对控制产生抵触情绪，研究如何诱发他们对控制的积极态度。

（5）要注意预测项目过程的发展趋势。事后及时发现偏差，不如在预见可能发生的偏差基础上采取预防措施，防患于未然。

（6）要有灵活性。项目的内外环境都会有变化。控制人员应事先准备有备用方案和措施。一招不灵，拿出另一招。

（7）要有重点。项目在进行中，千头万绪，不可能事事关照，时时关照。一定要抓住对实现项目目标有重大影响的关键问题和关键时点。在项目进度管理中，就要抓住里程碑。抓住重点，可大大提高控制工作的效率。抓住重点，还意味着把注意力集中在异常情况上。一般的正常情况无须多加关照。异常情况抓住了，就相当抓住了牛鼻子，抓住了关键。

（8）要便于项目干系人了解情况。向有关人员介绍情况，常常要使用数据、图表、文字说明、数学公式等。项目管理人员一定要保证这些手段直观、形象，一目了然。口头介绍时，要语言通俗、重点突出、简明扼要。

（9）要有全局观念。项目的各个方面都需要控制，进度、质量、费用、人力资源、合同等。特别要注意防止头疼医头，脚痛医脚。如在进度拖延时，不考虑其他后果，简单地靠增加投入来赶进度就不能算有全局观念。增加投入往往会损害费用控制目标。

11.4 项目成本管理

项目的成本管理主要是在批准的预算条件下，确保项目保质按期完成，其主要包括项目资源计划、项目成本估算、项目成本安排和项目成本控制。

11.4.1 项目成本计划

项目成本计划主要有资源计划、成本估算和成本预算三个子过程。

1．资源计划

项目中各项活动需要投入的资源多少以及资源投入的时间是项目经理事先要计划的，进行资源计划，必须利用如下事项作为依据。

（1）工作分解结构：项目工作分解结构已列出项目各组成部分需要何种资源，因此是资源计划的基本依据。

（2）范围说明书：范围说明书阐述了项目的必要性以及项目的各项目标。对此，在资源计划过程中都应当明确地给予考虑。

（3）后备资源说明书：进行资源计划时，必须从后备资源说明书中了解有哪些资源（人、设备、材料）可供使用。后备资源说明书内容多寡和详细程度因情况而异。

（4）组织策略：在资源计划期间必须考虑项目实施组织有关人员招聘，物资和设备

租用或采购的策略。

具体进行资源计划时，应当充分利用项目团队成员的技能和知识以及过去有类似资源要求的项目的历史资料。资源计划完成时要写出一份资源要求说明书，列出本项目要求使用的资源类型、数量以及工作分解结构各部分需要的数量。

2．成本估算

成本估算就是估算投入项目各活动所有资源的成本，并编制成本估算书。根据成本估算书可以进行项目内和项目间的比较。成本估算是对项目可能的结果进行定量估算，即为了达成项目既定目标，项目实施组织需要付出多少成本。进行成本估算之前必须要掌握如下几个方面的数据资料。

（1）资源要求：需要哪些资源以及多少资源。

（2）资源单价：只有知道了各种资源的单价，才能计算出项目的各种成本。

（3）时间估算：由于资金有时间价值，因此项目本身以及各项活动的各项时间参数就会对项目成本估算产生影响。

（4）成本编码：成本编码有项目实施组织用于报告一般日记账中财务资料的编码结构。项目成本估算必须记入正确的账目。

在进行成本估算时，可以利用如下数据、资料或手段作为工具。

（1）有关资源成本的资料。可以从下列来源取得这种资料：

- 项目档案。参与该项目的组织或项目团队成员保存的以前项目各种记录中有些很详细，完全可以用作成本估算。
- 市场上出售的各种成本估算数据库，如手册、书籍、磁盘或光盘等。
- 项目团队知识。有些项目团队成员可能会记得以前的实际成本或成本估算。

（2）建立参数模型。参数模型就是根据项目特点计算项目总成本的数学模型。如下情况使用参数模型比较可靠：

- 项目彼此类似。
- 建立模型时使用的历史资料准确。
- 模型可以适用不同规模。

（3）从下向上逐项估算。即先估算各工作块的成本，然后汇总出项目成本总和。由于项目划分成较小的部分，所以能够做出较准确的估算。

（4）类比估算。类比估算也叫从上向下估算，意思是利用以前类似项目的实际成本作为估算当前项目成本的基础。

（5）工具估算。例如，利用项目管理软件辅助成本估算。

3．编制成本计划

完成了成本估算，就可以编制成本计划，编制成本计划就是把整个项目的成本估算分配到各项目活动上去，进而确定测量项目实际执行情况的基准。编制时还要以项目工作分解结构和进度计划为依据。因为工作分解结构决定了项目哪些组成部分将发生成本，

而进度计划中则有此类组成部分的计划开始日期和预期结束日期，是按时间进程分配资金的依据。

11.4.2 软件成本估算方法

软件项目的规模估算历来是比较复杂的事，因为软件本身的复杂性、历史经验的缺乏、估算工具缺乏以及一些人为错误，导致软件项目的规模估算往往和实际情况相差甚远。因此，估算错误已被列入软件项目失败的四大原因之一。

软件工程师经常会被问到，编一个什么样的软件需要多长时间、多少资金。对于这个问题，不少人会犯难，原因有两个：一是用户的需求太不具体，二是自己缺乏一个科学的估算方法。本节介绍几种软件项目规模的估算方法。

先介绍一个衡量软件项目规模最常用的概念——代码行（Line Of Code，LOC），LOC指所有的可执行的源代码行数，包括可交付的工作控制语言（Job Control Language，JCL）语句、数据定义、数据类型声明、等价声明、输入/输出格式声明等。一行代码（1LOC）的价值和人月均代码行数可以体现一个软件生产组织的生产能力。组织可以根据对历史项目的审计来核算组织的单行代码价值。

例如，希赛 IT 教育研发中心统计发现，该中心每 1 万行 ASP .NET 源代码形成的源文件（aspx 文件）约为 250K。希赛在线教育平台项目的源文件大小为 3.75M，则可估算该项目源代码大约为 15.36 万行，该项目累计投入工作量为 240 人月，每人月成本为 10 000 元（包括人均工资、福利、办公成本公摊等），则该项目中 1LOC 的价值为：

$$(240\times 10\ 000)/153\ 600=15.625\ \text{元/LOC}$$

该项目的人月均代码行数为：

$$153\ 600/240=640\text{LOC/人月}$$

1．Delphi 法

Delphi 法是目前最流行的专家评估技术，在没有历史数据的情况下，这种方式适用于评定过去与将来、新技术与特定程序之间的差别，但专家“专”的程度以及对项目的理解程度是工作中的难点，尽管 Delphi 技术可以减少这种偏差，但通常在评定一个新软件的实际成本时用得不多，这种方式在决定其他模型的输入时特别有用。Delphi 法鼓励参加者根据问题进行相互讨论。这个技术要求有多种软件相关经验的人参与，互相说服对方。

Delphi 法的步骤如下：

（1）协调人向各专家提供项目规格和估算表格。

（2）协调人召集小组会的各专家讨论与规模相关的因素。

（3）各专家匿名填写迭代表格，如图 11-6 所示。

（4）协调人整理出一个估算总结，以迭代表的形式返回专家。

（5）协调人召集小组会，讨论较大的估算差异。

Delphi 法规模估算迭代表

项目名称：________

估算日期：________

估算者：________

估算轮次：________

结果：

代码行________　LOC；周期：____月；工作量：_____人月；成本_________元。

理由：

图 11-6　Delphi 法规模估算迭代表样例

（6）专家复查估算总结并在迭代表上提交另一个匿名估算。

（7）重复（4）～（6）步骤，直到最低和最高估算达到一致。

2．类比估算法

类比估算法适合评估一些与历史项目在应用领域、环境和复杂度相似的项目，通过新项目与历史项目的比较得到规模估算。类比法估算结果的精确度取决于历史项目数据的完整性和准确度，因此，用好类比法的前提条件之一是组织建立起较好的项目后评价与分析机制，对历史项目的数据分析是可信赖的。

类比估算法的基本步骤如下：

（1）整理出项目功能列表和实现每个功能的代码行。

（2）标识出每个功能列表与历史项目的相同点和不同点，特别要注意历史项目做得不够的地方。

（3）通过步骤（1）和（2）得出各个功能的估算值。

（4）产生规模估算。

软件项目中用类比估算法，往往还要解决可重用代码的估算问题。估算可重用代码量的最好办法就是由程序员或系统分析师详细地考查已存在的代码，估算出新项目可重用的代码中需重新设计的代码百分比、需重新编码或修改的代码百分比以及需重新测试的代码百分比。根据这三个百分比，可用下面的计算公式来计算等价新代码行：

等价代码行 = [(重新设计% +重新编码% +重新测试%)/3]×已有代码行

例如，希赛教育网视频点播系统有 10 000 行代码，假定 30%需要重新设计，50%需要重新编码，70%需要重新测试，那么其等价的代码行可以计算为：

$$[(30\% + 50\% + 70\%)/3\ ]\times 10\,000 = 5000 \text{ 等价代码行}$$

也就是说，重用这 10000 代码相当于编写 5000 代码行的工作量。

3．功能点估算法

功能点估算是在需求分析阶段基于系统功能的一种规模估算方法。通过研究初始应用需求来确定各种输入、输出、计算和数据库需求的数量和特性。通常的步骤如下：

（1）计算输入、输出、查询、主控文件和接口需求的数目。

（2）将这些数据进行加权相乘。如表 11-2 所示就是一个典型的权值表。

表 11-2　权值表举例

功 能 类 型	权　　值
输入	4
输出	5
查询	4
主控文件	10
接口	10

（3）估算者根据对复杂度的判断，总数可以用+25%、0 或–25%调整。

统计发现，对一个软件产品的开发，功能点对项目早期的规模估算很有帮助。然而，在了解产品越多后，功能点可以转换为软件规模测量更常用的 LOC 方法。

11.4.3　成本控制

项目成本控制的目的就是要保证各项工作在项目的预算范围内进行。成本控制的基础是事先就对项目进行项目预算。

由于成本、进度和技术三者密不可分，成本管理决不能脱离技术管理和进度管理独立存在，相反要在成本、技术、进度三者之间进行综合平衡。要实现这种全过程控制（事前、事中、事后）和全方位控制（成本、进度、技术），离不开反映及时、准确的动态信息的反馈系统。这一信息反馈系统主要是对成本、进度和技术进行跟踪报告，以便实行成本控制。

成本控制的基本方法是规定各部门定期上报其成本报告，再由控制部门对其进行成本审核，以保证各种支出的合法性，然后再将已经发生的成本与预算相比较，分析其是否超支，并采取相应的措施加以弥补。

成本控制最常用的方法是挣值法。挣值法实际上是一种分析目标实施与目标期望之间差异的方法。故而它又常被称为偏差分析法。挣值法通过测量和计算已完成的工作的预算成本与已完成工作的实际成本和计划工作的预算成本得到有关计划实施的进度和成本偏差，而达到判断项目预算和进度计划执行情况的目的。因此它的独特之处在于以预算和成本来衡量工程的进度。挣值法取名正是因为这种分析方法中用到的一个关键数值

——挣值（即是已完成工作预算），而以其来命名的。

1．挣值法的三个基本参数

（1）计划工作量的预算成本（Budgeted Cost for work Scheduled，BCWS）。BCWS 是指项目实施过程中某阶段计划要求完成的工作量所需的预算工时（或成本）。计算公式为：

BCWS＝ 计划工作量×预算定额

BCWS 主要是反映进度计划应当完成的工作量，而不是反映应消耗的工时或成本。在有些文献中，把 BCWS 称为 PV（Plan Value，计划值）。

（2）已完成工作量的实际成本（Actual Cost for Work Performed，ACWP）。ACWP 是指项目实施过程中某阶段实际完成的工作量所消耗的工时（或成本）。ACWP 主要反映项目执行的实际消耗指标。在有些文献中，把 ACWP 简称为 AC（Actual Cost，实际成本）。

（3）已完工作量的预算成本（Budgeted Cost for Work Performed，BCWP）。BCWP 是指项目实施过程中某阶段实际完成工作量及按预算定额计算出来的工时（或成本），即挣值（Earned Value，EV）。BCWP 的计算公式为：

BCWP＝ 已完成工作量×预算定额

2．挣值法的 4 个评价指标

（1）成本偏差（Cost Variance，CV）。CV 是指检查期间 BCWP 与 ACWP 之间的差异，计算公式为：

CV＝BCWP–ACWP

（2）进度偏差（Schedule Variance，SV）。SV 是指检查日期 BCWP 与 BCWS 之间的差异。其计算公式为：

SV＝BCWP–BCWS

（3）成本绩效指数（Cost Performed Index，CPI）。CPI 是指项目挣值与实际成本值之比（或工时值之比）。计算公式为：

CPI＝BCWP/ACWP

当 CPI>1，表示低于预算，即实际成本低于预算成本。

当 CPI<1，表示超出预算，即实际成本高于预算成本。

当 CPI=1，表示实际成本与预算成本吻合。

（4）进度绩效指数（Schedul Performed Index，SPI）。SPI 是指项目挣值与计划值之比，即：

SPI＝BCWP/BCWS

当 SPI>1，表示进度提前，即实际进度比计划进度快。

当 SPI<1，表示进度延误，即实际进度比计划进度慢。

当 SPI=1，表示实际进度等于计划进度。

11.5 项目质量管理

根据 ISO 9000：2000 标准定义，所谓质量，是指一组固有特性满足要求的程度。质量管理是指在质量方面指挥和控制组织的协调的活动。在质量方面的指挥和控制活动，通常包括制定质量方针和质量目标以及质量策划、质量控制、质量保证和质量改进。可见，质量管理是质量管理主体围绕着使产品质量能满足不断更新的质量要求，而开展的策划、组织、计划、实施、检查和监督、审核等所有管理活动的总和。

11.5.1 质量管理计划

项目质量管理计划就是确定项目应当采用哪些质量标准以及如何达到。质量计划是保证项目成功的过程之一，应当同项目其他计划过程结合起来。事先不计划，指望在项目实施过程中靠检查和督促来保证项目质量是不行的。

质量计划过程应以下列事项为依据。

（1）质量方针：质量方针就是项目实施组织领导层就质量问题明确阐明的所有打算和努力方向。

（2）范围说明：项目的范围说明不但规定了主要的项目成果，而且也规定了项目的目标，是质量计划的关键依据。

（3）成果说明：成果说明是对范围说明书中的项目成果的进一步说明，经常包括有技术问题以及可能影响质量计划的其他问题的细节。

（4）标准和规范：项目管理团队必须考虑对该项目可能产生影响的任何应用领域的专用标准和规范。

（5）其他过程的结果：除了范围说明和成果说明之外，其他知识领域过程的结果，也可能同质量计划有关。

质量计划是质量计划的主要成果，其内容是对特定的项目、产品、过程或合同，规定由谁及何时，应使用哪些程序和相关资源的文件项目的质量计划是针对具体项目的要求，以及应重点控制的环节所编制的对设计、采购、项目实施、检验等质量环的质量控制方案。质量计划往往并不是单独一个文件，而是由一系列文件所组成。项目开始时，应从总体考虑，编制一个保证项目质量的计划性的质量计划，如质量管理计划；随着项目的进展，编制相应的各阶段较详细的质量计划，如项目操作规范。项目质量计划的格式和详细程度并无统一规定，但应与用户的要求、供方的操作方式和活动的复杂程度等相适应。计划应尽可能简明。

质量计划应明确指出所开展的质量活动，并直接指出或间接指出（通过相应程序或其他文件）如何实施所要求的活动。其内容包括：

（1）需达到的质量目标，包括项目总质量目标和具体目标。

（2）质量管理工作流程，可以用流程图等形式展示过程的各项活动。

（3）在项目的各个不同阶段，职责、权限和资源的具体分配。

（4）项目实施中需采用的具体的书面程序和指导书。

（5）有关阶段适用的试验、检查、检验和评审大纲。

（6）达到质量目标的测量方法。

（7）随项目的进展而修改和完善质量计划的程序。

（8）为达到项目质量目标必须采取的其他措施，如更新检验技术、研究新的工艺方法和设备、用户的监督、验证等。

这些内容可能包含在不同的质量计划文件之中。

11.5.2　质量控制和质量保证

项目质量管理的两个主要活动是质量控制和质量保证，这两个活动既有区别也有联系。

1．质量控制

质量控制是质量管理的一部分，致力于满足质量要求。质量控制的目标就是确保项目质量能满足有关方面所提出的质量要求（如适用性、可靠性、安全性等）。质量控制的范围涉及到项目质量形成全过程的各个环节。项目质量受到质量环各阶段质量活动的直接影响，任一环节的工作没有做好，都会使项目质量受到损害而不能满足质量要求。质量环的各阶段是由项目的特性所决定的，根据项目形成的工作流程，由掌握了必需的技术和技能的人员进行一系列有计划、有组织的活动，使质量要求转化为满足质量要求的项目或产品并完好地交付给用户，还应根据项目的具体情况进行用后服务，这是一个完整的质量循环。为了保证项目质量，这些技术计划必须在受控状态下进行。

质量控制的工作内容包括了作业技术和活动，即包括专业技术和管理技术两方面。质量控制应贯彻预防为主与检验把关相结合的原则，在项目形成的每一个阶段和环节，即质量环的每一阶段，都应对影响其工作质量的人、机、料、法、环因素进行控制，并对质量活动的成果进行分阶段验证，以便及时发现问题，查明原因，采取措施，防止类似问题重复发生，并使问题在早期得到解决，减少经济损失。为使每项质量活动都能有效，质量控制对干什么、为何干、如何干、由谁干、何时干、何地干等问题应做出规定，并对实际质量活动进行监控。项目的进行是一个动态过程，所以，围绕项目的质量控制也具有动态性。为了掌握项目随着时间的变化而变化的状态，应采用动态控制的方法和技术进行质量控制工作。

质量控制的依据是工作结果和质量管理计划、实施说明以及核对表。质量控制的工具和技术有检查、控制图和排列图等。检查包括测量、审查和试验质量控制结果，以便判断这些结果是否符合要求。质量控制的结果应当有项目质量的改进、工作结果验收或不予接受、返工和过程调整等。

2．质量保证

随着技术的发展，项目越来越复杂，对项目质量要求也越来越高，项目的有些性能已不能通过检验来鉴定。就这些项目来说，用户为了确信项目实施者所完成的项目达到了所规定的质量要求，就要求项目实施者提供项目设计、实施等各个环节的主要质量活动确实做好，且能提供合格项目的证据，这就是用户提出的质量保证要求。针对用户提出的质量保证要求，项目实施者就应开展外部质量保证活动，就应对用户提出的设计、项目实施等全过程中的某些环节的活动提供必要的证据，以使用户放心。

今天，质量保证的内涵已不是单纯的为了保证质量，保证质量是质量控制的任务，而是以保证质量为基础，进一步引申到值得客户信任这一根本目的上。为此，项目实施者应加强质量管理，完善质量体系，对项目有一套完善的质量控制方案、办法，并认真贯彻执行，对实施过程及成果进行分阶段验证，以确保其有效性。

在此基础上，项目实施者应有计划、有步骤地采取各种活动和措施，使用户能了解其实力、业绩、管理水平、技术水平以及对项目在设计、实施各阶段主要质量控制活动和内部质量保证活动的有效性，使对方建立信心，相信完成的项目能达到所规定的质量要求。所以，质量保证的主要工作是促使完善质量控制，以便准备好客观证据，并根据对方的要求有计划、有步骤地开展提供证据的活动。

质量保证的依据是质量管理计划、实施说明和质量控制测量的结果。质量保证的工具和技术有质量计划和检查。质量保证应当取得的结果是项目质量的改进。质量改进应该包括采取行动提高项目的效率和效果，为项目的所有干系人增加收益。

11.5.3 软件质量管理概述

软件是逻辑产品，其质量属性有不同的特点，通常，主要从下面 6 个方面来衡量一个软件的质量。

（1）性能。是指系统的响应能力，即要经过多长时间才能对某个事件做出响应，或者在某段事件内系统所能处理的事件的个数。经常用单位事件内所处理事务的数量或系统完成某个事务处理所需的时间来对性能进行定量的表示。性能测试经常要使用基准测试程序（用以测量性能指标的特定事务集或工作量环境）。

（2）可靠性。是软件系统在应用或系统错误面前，在意外或错误使用的情况下维持软件系统的功能特性的基本能力。可靠性是最重要的软件特性，通常用它衡量在规定的条件和时间内，软件完成规定功能的能力。可靠性通常用 MTTF 和 MTBF 来衡量。在失效率为常数和修复时间很短的情况下，MTTF 和 MTBF 几乎相等。

可靠性可以分为容错和健壮性两个方面。其中，容错的目的是在错误发生时确保系统正确的行为，并进行内部“修复”。例如在一个分布式软件系统中失去了一个与远程构件的连接，接下来恢复了连接。在修复这样的错误之后，软件系统可以重新或重复执行进程间的操作直到错误再次发生；健壮性是指保护应用程序不受错误使用和错误输入的

影响，在遇到意外错误事件时确保应用系统处于已经定义好的状态。值得注意的是，和容错相比，健壮性并不是说在错误发生时软件可以继续运行，它只能保证软件按照某种已经定义好的方式终止执行。软件体系结构对软件系统的可靠性有巨大的影响。例如软件体系结构通过在应用程序内部包含冗余，或集成监控构件和异常处理，来支持可靠性。

（3）可用性。是系统能够正常运行的时间比例。经常用两次故障之间的时间长度或在出现故障时系统能够恢复正常的速度来表示。

（4）安全性。是指系统在向合法用户提供服务的同时能够阻止非授权用户使用的企图或拒绝服务的能力。安全性是根据系统可能受到的安全威胁的类型来分类的。安全性又可划分为机密性、完整性、不可否认性及可控性等特性。其中，机密性保证信息不泄露给未授权的用户、实体或过程；完整性保证信息的完整和准确，防止信息被非法修改；可控性保证对信息的传播及内容具有控制的能力，防止为非法者所用。

（5）可修改性。是指能够快速地以较高的性能价格比对系统进行变更的能力。通常以某些具体的变更为基准，通过考察这些变更的代价衡量可修改性。可修改性包含可维护性、可扩展性、结构重组、可移植性等 4 个方面。

可维护性主要体现在问题的修复上：在错误发生后“修复”软件系统。为可维护性做好准备的软件体系结构往往能做局部性的修改并能使对其他构件的负面影响最小化。

可扩展性关注的是使用新特性来扩展软件系统，以及使用改进版本来替换构件并删除不需要或不必要的特性和构件。为了实现可扩展性，软件系统需要松散耦合的构件。其目标是实现一种体系结构，它能使开发人员在不影响构件客户的情况下替换构件。支持把新构件集成到现有的体系结构中也是必要的。

结构重组处理的是重新组织软件系统的构件及构件间的关系，例如通过将构件移动到一个不同的子系统而改变它的位置。为了支持结构重组，软件系统需要精心设计构件之间的关系。理想情况下，它们允许开发人员在不影响实现的主体部分的情况下灵活地配置构件。

可移植性使软件系统适用于多种硬件平台、用户界面、操作系统、编程语言或编译器。为了实现可移植，需要按照硬件无关的方式组织软件系统，其他软件系统和环境被提取出。可移植性是系统能够在不同计算环境下运行的能力。这些环境可能是硬件、软件，也可能是两者的结合。在关于某个特定计算环境的所有假设都集中在一个构件中时，系统是可移植的。如果移植到新的系统需要做些更改，则可移植性就是一种特殊的可修改性。

（6）功能性。是系统所能完成所期望的工作的能力。一项任务的完成需要系统中许多或大多数构件的相互协作。

与硬件等其他产品相比，软件产品的质量有以下特点：

（1）对于不同类型的软件产品，其所考察的 6 个质量属性的侧重点不一样。例如，对于实时系统而言，性能和效率是需要考虑的首要因素；而对一个公安身份证系统来说，

安全性则是第一位的。

（2）软件产品的质量属性很难量化，也没有相应的国际标准、国家标准或行业标准。对软件产品而言，无法确定诸如“合格率”、“一次通过率”、PPM、“寿命”之类的质量目标。当前通用的方法是使用每千行的缺陷数来对软件质量进行度量，但缺陷的等级、种类、性质、影响不同，我们不能说每千行缺陷数量小的软件，一定比该数量大的软件质量更好。

（3）因为没有一个“通用”的标准来衡量软件质量的好坏，所以，软件产品的质量没有绝对的合格/不合格界限。因此，一般来说，凡是满足了用户需求的软件，就是好的软件。

（4）软件产品不可能做到“零缺陷”，特别对大中型软件而言更是如此。虽然我们可以通过软件测试尽可能地发现和改正软件中的缺陷，但对软件的测试不可能穷尽所有情况。

11.5.4 软件质量保证体系

质量保证是为保证产品和服务充分满足消费者要求的质量而进行的有计划、有组织的活动。质量保证是面向消费者的活动，是为了使产品实现用户要求的功能，站在用户立场上来掌握产品质量的。质量保证的目标是为管理层提供为获知产品质量信息所需的数据，从而获得产品质量是否符合预定目标的认识和信心。软件质量保证（Software Quality Assurance，SQA）活动是确保软件产品在软件生存期所有阶段的质量的活动，即为了确定、达到和维护需要的软件质量而进行的所有有计划、有系统的管理活动。

SQA 由各项任务构成，这些任务的参与者有两种人：软件开发人员和质量保证人员。前者负责技术工作，后者负责质量保证的计划、监督、记录、分析及报告工作。

软件开发人员通过采用可靠的技术方法和措施，进行正式的技术评审，执行软件测试来保证软件产品的质量。SQA 人员则辅助软件开发组得到高质量的最终产品。

1．SQA 人员的素质

（1）SQA 人员要有较强的沟通能力。从实施 SQA 的目的中可以看出，SQA 人员不在项目中，是独立于软件项目的第三方，但他要了解项目的开发过程和进度，捕捉到项目中不符合要求的问题，这就要求 SQA 人员能够深入项目，与软件开发人员保持良好的沟通，这样才能及时获得真实的项目情况和准确的项目数据。

（2）SQA 人员要熟悉软件工程过程。作为 SQA 人员，既然要确保项目组制定的计划、标准和规程，要符合项目组要求，那么 SQA 人员自己要了解软件开发过程，以及组织内部已有的软件过程规范。

（3）SQA 人员要有较强的计划性。SQA 人员不但要监督软件项目组编写计划，而且 SQA 人员自身的工作也要有计划，并且能够按照计划开展工作。

（4）SQA 人员要能应对繁杂的工作。作为 SQA 人员，在对项目进行跟踪时，要对

项目组的很多工作产品进行审计，而且会参与项目组中的多种活动。同时一个 SQA 人员还有可能会面对多个项目组，所以任务相对繁杂细碎，这就要求 SQA 人员在处理这些事物的时候要耐心细致。

（5）SQA 人员要客观，有责任心。作为第三方对项目过程进行监督，SQA 人员要能保持自己的客观性，不能一味讨好项目经理，也不能成为项目组中的宪兵，否则会影响工作的开展。对于项目组中多次协调解决不了的问题，能够向项目的高层经理进言，完成 SQA 的使命。

此外，一个好的 SQA 人员还应该在软件开发过程中作为开发人员或测试人员参与过一个或多个环节，这样他们才能在过程监督中比较准确地抓住重点，同时他们的意见和提出的解决办法也会更贴近项目组，容易被项目组接受。

2．SQA 活动

1993 年，美国卡耐基 • 梅隆大学软件工程研究所推荐了一组有关质量保证的计划、监督、记录、分析及报告的 SQA 活动。这些活动将由一个独立的 SQA 小组执行。

（1）制定 SQA 计划。SQA 计划在制定项目计划时制定，由相关部门审定。它规定了软件开发小组和质量保证小组需要执行的质量保证活动，其要点包括：需要进行哪些评价；需要进行哪些审计和评审；项目采用的标准；错误报告的要求和跟踪过程；SQA 小组应产生哪些文档；为软件项目组提供的反馈数量等。

（2）参与开发该软件项目的软件过程描述。软件开发小组为将要开展的工作选择软件过程，SQA 小组则要评审过程说明，以保证该过程与组织政策、内部的软件标准、外界所制定的标准（例如，CMM 等）以及软件项目计划的其他部分相符。

（3）评审。评审各项软件工程活动，核实其是否符合已定义的软件过程。SQA 小组识别、记录和跟踪所有偏离过程的偏差，核实其是否已经改正。

（4）审计。审计指定的软件工作产品，核实其是否符合已定义的软件过程中的相应部分。SQA 小组对选出的产品进行评审，识别、记录和跟踪出现的偏差，核实其是否已经改正，定期向项目负责人报告工作结果。

（5）记录并处理偏差。确保软件工作及工作产品中的偏差已被记录在案，并根据预定规程进行处理。偏差可能出现在项目计划、过程描述、采用的标准或技术工作产品中。

（6）报告。记录所有不符合部分，并向上级管理部门报告。跟踪不符合的部分直到问题得到解决。

除了进行上述活动外，SQA 小组还需要协调变更的控制与管理，并帮助收集和分析软件度量的信息。

3．SQA 工作内容

SQA 的工作内容主要包括以下 5 类：

（1）与 SQA 计划直接相关的工作。SQA 人员在项目早期要根据项目计划制定与其对应的 SQA 计划，定义出各阶段的检查重点，标识出检查、审计的工作产品对象，以

及在每个阶段SQA的输出产品。定义越详细，对于SQA今后的工作的指导性就会越强，同时也便于软件项目经理和SQA组长对其工作的监督。编写完SQA计划后要组织SQA计划的评审，并形成评审报告，把通过评审的SQA计划发送给软件项目经理、项目开发人员及所有相关人员。

（2）参与项目的阶段性评审和审计。在SQA计划中通常已经根据项目计划定义了与项目阶段相应的阶段检查，包括参加项目在本阶段的评审和对其阶段产品的审计。对于阶段产品的审计通常是检查其阶段产品是否按计划按规程输出并内容完整，这里的规程包括组织内部统一的规程也包括项目组内自己定义的规程。但是SQA人员对于阶段产品内容的正确性一般不负责任检查，对于内容的正确性通常交由项目中的评审来完成。SQA人员参与评审是从保证评审过程有效性方面入手，如参与评审的人是否具备一定资格、是否规定的人员都参加了评审、评审中对被评审的对象的每个部分都进行了评审、并给出了明确的结论等等。

（3）对项目日常活动与规程的符合性进行检查。这部分的工作内容是SQA人员的日常工作内容。由于SQA人员独立于项目组，如果只是参与阶段性的检查和审计，则很难及时反映项目组的工作过程，所以SQA人员也要在两个阶段点之间设置若干小的跟踪点，来监督项目的进行情况，以便能及时反映出项目组中存在的问题，并对其进行跟踪。如果只在阶段点进行检查和审计，即便发现了问题也难免过于滞后，不符合尽早发现问题、把问题控制在最小的范围之内的整体目标。

（4）对配置管理工作的检查和审计。SQA人员要对项目过程中的配置管理工作是否按照项目最初制定的配置管理计划进行监督，包括配置管理人员是否定期进行该方面的工作、是否所有人得到的都是开发过程产品的有效版本。

（5）跟踪问题的解决情况。对于评审中发现的问题和项目日常工作中发现的问题，SQA人员要进行跟踪，直至解决。对于在项目组内可以解决的问题就在项目组内部解决，对于在项目组内部无法解决的问题，或是在项目组中跟催多次也没有得到解决的问题，可以利用其独立汇报的渠道报告给高层经理。

4. SQA计划

具体项目开发组根据公司质量体系制订质量活动计划并形成SQA计划，以保证开发组能正确理解质量体系并能遵照执行。SQA计划的目的是计划出哪些是需要被跟踪的工作，并建立文档。此文档可以作为SQA代表的工作指南，帮助项目经理确保所有工作按计划完成。因此，SQA计划是由技术部门经理或项目主管填写完成，而SQA代表又籍此来帮助技术部门经理和项目主管。在起草完SQA计划初始稿后，应该发给SQA代表，从而使他能够参与制定最初的SQA计划。

SQA计划的关键要素应包括：

（1）任务管理：任务管理程序、质量计划、客户满意调查等；

（2）人员管理：人事政策与程序、培训与培养、指导工作；

（3）SQA 计划管理：质量政策、质量组织、计划的聚焦点。

下面，参考有关标准，给出 SQA 计划必须具有的内容。

1 引言

1.1 目的

指出特定的 SQA 计划的具体目的，给出该计划所针对的软件项目（及其所属的各个子项目）的名称和用途。

1.2 定义和缩写词

列出计划正文中需要解释的而在 GB/T 11457 中尚未包含的术语的定义，必要时，还要给出这些定义的英文单词及其缩写词。

1.3 参考资料

列出计划正文中所引用资料的名称、代号、编号、出版机构和出版年月。

2 管理

描述负责 SQA 的机构，任务及其有关的职责。

2.1 机构

描述与 SQA 有关的机构的组成。还必须清楚地描述来自项目委托单位、项目承办单位、软件开发单位或用户中负责软件质量保证的各个成员在机构中的相互关系。

2.2 任务

描述计划所涉及的软件生存周期中有关阶段的任务，特别要把重点放在描述这些阶段所应进行的软件质量保证活动上。

2.3 职责

指明 SQA 计划中规定的每一个任务的负责单位或成员的责任。

3 文档

必须列出在该软件的开发、验证与确认以及使用与维护等阶段中需要编制的文档，并描述对文档进行评审与检查的准则。

3.1 基本文档

为了确保软件的实现满足需求，至少需要下列基本文档：软件需求规格说明书、软件设计说明书、软件验证与确认计划、软件验证和确认报告、用户文档、其他文档。

除基本文档外，还应包括下列文档：项目实施计划、项目进展报表、项目开发各个阶段的评审报表、项目开发总结。

有关这些文档的编制，请参考本书第 10 章。

4 标准、条例和约定

列出软件开发过程中要用到的标准、条例和约定，并列出监督和保证书执行的措施。

5 评审和检查

规定所要进行的技术和管理两方面的评审和检查工作，并编制或引用有关的评审和检查以及通过与否的技术准则。至少要进行下列各项评审和检查工作：

5.1 软件需求评审

在软件需求分析阶段结束后必须进行软件需求评审，以确保在软件需求规格说明书中所规定的各项需求的合适性。

5.2 概要设计评审

在软件概要设计结束后必须进行概要设计评审，以评价软件设计说明书中所描述的软件概要设计的总体结构、外部接口、主要部件功能分配、全局数据结构以及各主要部件之间的接口等方面的合适性。

5.3 详细设计评审

在软件详细设计阶段结束后必须进行详细设计评审，以确定软件设计说明书中所描述的详细设计在功能、算法和过程描述等方面的合适性。

5.4 软件验证与确认评审

在制订软件验证与确认计划之后要对它进行评审，以评价软件验证与确认计划中所规定的验证与确认方法的合适性与完整性。

5.5 功能检查

在软件释放前，要对软件进行功能检查，以确认已经满足在软件需求规格说明书中规定的所有需求。

5.6 物理检查

在验收软件前，要对软件进行物理检查，以验证程序和文档已经一致并已做好了交付的准备。

5.7 综合检查

在软件验收时，要允许用户或用户所委托的专家对所要验收的软件进行设计抽样的综合检查，以验证代码和设计文档的一致性、接口规格说明之间的一致性（硬件和软件）、设计实现和功能需求的一致性、功能需求和测试描述的一致性。

5.8 管理评审

要对计划的执行情况定期（或按阶段）进行管理评审；这些评审必须由独立于被评审单位的机构或授权的第三方主持进行。

6 软件配置管理

编制有关软件配置管理的条款，或引用按照 GB/T 12505 单独制订的文档。在这些条款或文档中，必须规定用于标识软件产品、控制和实现软件的修改、记录和报告修改实现的状态以及评审和检查配置管理工作等 4 方面的活动。还必须规定用以维护和存储软件受控版本的方法和设施；必须规定对所发现的软件问题进行报告、追踪和解决的步骤，并指出实现报告、追踪和解决软件问题的机构及其职责。

7 工具、技术和方法

指明用以支持特定软件项目质量保证工作的工具、技术和方法，指出它们的目的，描述它们的用途。

8 媒体控制

指出保护计算机程序物理媒体的方法和设施，以免非法存取、意外损坏或自然老化。

9 对供货单位的控制

供货单位包括项目承办单位、软件销售单位、软件开发单位或软件子开发单位。规定对这些供货单位进行控制的规程，从而保证项目承办单位从软件销售单位购买的、其他开发单位（或子开发单位）开发的或从开发（或子开发）单位现存软件库中选用的软件能满足规定的需求。

10 记录的收集、维护和保存

指明需要保存的软件质量保证活动的记录，并指出用于汇总、保护和维护这些记录的方法和设施，并指明要保存的期限。

11.5.5　软件质量保证的实施

SQA 的实施需要整个 SQA 部门齐心协力，需要与用户合作，需要得到组织高层领导的支持，需要得到软件开发人员的理解。

在 SQA 的实施中，主要采用 PDCA 循环（戴明环），其中的 P 为计划，D 为实施（执行），C 为检查，A 为处理（行动）。

第一阶段是计划，包括方针、目标、活动计划、管理项目等，设定适合于被开发软件的评测检查项目，与此同时，还要研讨实现质量目标的方法或手段。

第二阶段是实施，即按照计划的要求去做，在开发标准和质量评价准则的指导下，制作高质量的规格说明书和程序。在接受质量检查之前要先做自我检查。

第三阶段是检查，检查是否按规定的要求去做，哪些做对了，哪些没有做对，哪些有效果，哪些没有效果，并找出异常情况的原因。

第四阶段是处理，就是说，要把成功的经验肯定下来，变成标准。以后就按照这个标准去做。失败的教训也要加以总结，使它成为标准，防止以后再发生。没有解决的遗留问题反映到下一个循环中去。

计划、实施、检查、处理这个过程，不断反复进行，一个循环接着另一个循环，每一次都赋予新的内容。

1．正式技术评审

在软件开发的某一阶段中出现的错误，如果得不到及时纠正，就会传播到开发的后续阶段中去，并在后续阶段中引出更多的错误。实践证明，提交给测试阶段的程序中包

含的错误越多，经过同样时间的测试后，程序中仍然潜伏的错误也越多。所以必须在开发时期的每个阶段，特别是设计阶段结束时都要进行严格的技术评审，尽量不让错误传播到下一个阶段。

软件技术评审是软件工程师进行的一种质量保证活动，其目标如下：

（1）在软件的任何一种表示形式中发现功能、逻辑或实现的错误。

（2）证实经过评审的软件的确满足需求。

（3）保证软件的表示符合预定义的标准。

（4）得到以一种一致的方式开发的软件。

（5）使项目更易于管理。

另外，由于正式技术评审的进行使大量人员对软件系统中原本并不熟悉的部分更为了解，因此，还起到了提高项目连续性和培训后备人员的作用。正式技术评审包括“走查”（walkthrough）、“审查”（inspection）、“轮查”（Round-robin Review）和其他的软件评审技术。每次正式技术评审都以会议形式进行，只有在很好地计划、控制和参与的情况下，正式技术评审才有可能获得成功。

不管选择哪一种正式技术评审的形式，每次评审会议都需遵循以下约束：

（1）每次会议的参加人数为 3～5 人。

（2）会前应做好准备，但每个人的工作量不应超过 2 小时。

（3）每次会议的时间不应超过 2 小时。

在上述约束之下，显然正式技术评审关注的应是整个软件的某一特定（且较小）的部分。例如，不是对整个设计评审，而是逐个模块走查，或走查模块的一部分。通过缩小关注的范围，更容易发现错误。

正式技术评审的关注点集中于某个工作产品，即软件的某一部分（如部分需求规格说明、一个模块的详细设计、一个模块的源程序清单）。评审会议由评审负责人主持，所有评审人员和开发人员参加。正式技术评审首先讨论日程安排，然后让开发人员“遍历”其工作产品，做简单介绍。评审人员根据他们事先的准备提出问题。当问题被确认或错误被发现，记录员要将其一一记录下来。评审结束时，所有正式技术评审的参加者必须作出决定：

（1）接受该工作产品，不再做进一步的修改。

（2）由于该工作产品错误严重，拒绝接受（错误改正后必须再次进行评审）。

（3）暂时接受该工作产品（发现必须改正的微小错误，但不必再次进行评审）。

当决定之后，正式技术评审的所有参加者都必须签名，以表明他参加了会议，并同意评审组的决定。

进行正式技术评审之前，必须建立评审指南，分发给所有评审者，并得到大家的认可，然后才能依照该指南进行评审。下面是正式技术评审指南中必须包含的内容：

（1）评审产品而不是评审开发者。正式技术评审涉及到别人和自我，如果进行得恰

当，正式技术评审可以使所有参与者体会到成就感；但如果进行得不恰当，则可能陷入一种审问得气氛之中。评审人员应该要温和地指出错误，会议得气氛应该是轻松的和建设性的。

（2）制定日程，并且遵守日程。各种类型的会议都具有一个主要缺点，就是放任自流。正式技术评审必须保证不要离题和按照计划进行。

（3）限制争论和辩驳。在评审者提出问题时，未必所有人都认同该问题的严重性。不要花时间争论这一问题，这样的问题应该被记录在案，留到会后进一步讨论。

（4）对各个问题都发表见解，但是不要试图解决所有记录的问题。正式技术评审不是一个问题解决会议，问题的解决通常由开发者自己或者在其他人的帮助下来完成。问题的解决应该放到评审会议之后进行。

（5）做好书面笔记。有时候让记录员在黑板上做笔记是个好主意，这样在记录员记录信息时，其他的评审人员可以推敲措辞，并确定问题的有限次序。

（6）限制参与者人数，并坚持先做准备。将正式技术评审涉及的人员数量保持在最小的必须值上，所有参与评审的人员都必须事先做好准备。

（7）为每个可能要评审的工作产品建立一个检查表。检查表能帮助会议主持者组织正式技术评审会议，并帮助每个评审人员将注意力集中在重要问题上。

（8）对正式技术评审分配资源和时间。为了让评审有效，应该将评审作为软件工程过程中的任务加以调度，而且要为由评审结果导致的修改活动分配时间。

（9）对所有评审者进行培训。为了提高效率，所有参加评审的人员都应该接受某种正式培训。

（10）评审以前所做的评审。听取汇报对发现评审过程本身的问题十分有益，最早被评审的工作产品本身可能就会成为评审指南。

2．需要考虑的问题

（1）要考虑 SQA 人员的素质。SQA 人员的责任是审查软件设计、开发人员的活动，验证他们是否将选定的标准、方法和规程应用到活动中去，因此，SQA 工作的有效执行需要 SQA 人员掌握专业的技术，例如，质量控制知识、统计学知识等。

（2）SQA 人员的经验对任务的实现同样重要。应该选择那些经验丰富的人来做 SQA，同时为 SQA 人员进行专门的培训，以使他们能够胜任这项工作。

（3）组织应当建立文档化的开发标准和规程，使 SQA 人员在工作时有一个依据、判断的标准。如果没有这些标准，SQA 人员就无法准确地判断开发活动中的问题，容易引发不必要的争论。

（4）高级管理者必须重视软件质量保证活动。在一些组织的软件生产过程中，高级管理者不重视软件质量保证活动，对 SQA 人员发现的问题不及时处理。如此一来，软件质量保证就流于形式，很难发挥它应有的作用。

（5）SQA 人员在工作过程中一定要抓住问题的重点与本质，不要陷入对细节的争论

之中。SQA 人员应集中审查定义的软件过程是否得到了实现，及时纠正那些疏漏或执行得不完全的步骤，以此来保证软件产品的质量。

此外，做好 SQA 工作还应该有一个计划，用以规定 SQA 活动的目标，执行审查所参照的标准和处理的方式。对于一般性项目，可采用通用的 SQA 计划，而对于那些有着特殊质量要求的项目，则必须根据项目自身的特点制定专门的计划。

总之，SQA 是软件过程中的独立审查活动，它从一个侧面反映了现行软件过程能力的成熟度水平。SQA 活动是贯穿整个软件过程的，那种到编码之后才开始关心质量的做法是极其错误的。

11.5.6 全面质量管理

菲根堡姆于 1961 年在其《全面质量管理》一书中首先提出了全面质量管理（Total Quality Management，TQM）的概念：“全面质量管理是为了能够在最经济的水平上，并考虑到充分满足用户要求的条件下进行市场研究、设计、生产和服务，把组织内各部门研制质量、维持质量和提高质量的活动构成为一体的一种有效体系”。

TQM 即为全员、全过程、全方位的质量管理，力求全面提高经济效益。包括以下基本特点。

（1）全员参加：意味着质量控制由少数质量管理人员扩展到组织的所有人员。

（2）全过程：将质量控制从质量检验和统计质量控制扩展到整个产品寿命周期。

（3）全面运用一切有效方法：是指应用一切可以运用的方法，而不仅仅是数理统计法。

（4）全面控制质量因素：意味着把影响质量的人、设备、技术、方法、测试手段、软硬件环境等全部予以控制，以确保质量。

大多数经营管理者认为，TQM 的核心是强调一致性，克服随意性，消除差错，使用户得到全面的满足，它强调为了取得真正的经济效益，管理必须“始于识别用户的需要，终于满足用户的需要”。TQM 就是为了实现这一目标而指导人、设备、信息的协调活动。因此，TQM 可以归纳为两大基本原则：

（1）以满足用户需求为导向，不断改善，最终达到用户的全面满足。

（2）以全员参与为基础，进行全过程的质量控制。

1. 核心思想

TQM 在我国也得到一定的发展，有关专家总结实践中经验，提出了“三全一多样”的观点。即认为推行 TQM，必须要满足“三全一多样”的基本要求。

（1）全过程的质量管理。任何产品或服务的质量，都有一个产生、形成和实现的过程。为了保证和提高产品质量，就必须把影响质量的所有环节和因素都控制起来。为此，全过程的质量管理包括了从市场调研、产品的设计开发、测试，到销售、维护等全部有关过程的质量管理。因此，全面质量管理强调产品开发必须体现以预防为主、不断改进，

以及为用户服务的思想。

（2）全员的质量管理。产品和/或服务质量是组织各个部门、各环节工作质量的综合反映。组织中任何一个环节，任何一个人的工作质量都会不同程度地直接或间接地影响着产品质量或服务质量。因此，在组织的生产过程中，要灌输“产品质量，人人有责”的思想。

为此，必须抓好全员的质量教育和培训，制订各部门、各级各类人员的质量责任制，形成一个高效、协调、严密的质量管理工作的系统。最后，要开展多种形式的群众性质量管理活动，充分发挥广大职工的聪明才智和当家作主的进取精神。

（3）全组织的质量管理。全组织的质量管理可以从纵横两个方面来加以理解。从纵向的组织管理角度来看，质量目标的实现有赖于组织的上层、中层、基层管理乃至一线员工的通力协作，其中尤以高层管理能否全力以赴起着决定性的作用。从组织职能间的横向配合来看，要保证和提高产品质量必须使组织研制、维持和改进质量的所有活动构成为一个有效的整体。为了有效地进行 TQM，必须加强各部门之间的组织协调，并且为了从组织上、制度上保证组织长期稳定地生产出符合规定要求、满足用户期望的产品，最终必须要建立起全组织的质量管理体系，使组织的所有研制、维持和改进质量的活动构成为一个有效的整体。建立和健全组织质量管理体系，是 TQM 深化发展的重要标志。可见，全组织的质量管理就是要“以质量为中心，领导重视、组织落实、体系完善。”

（4）多方法的质量管理。影响产品质量和服务质量的因素也越来越复杂，必须根据不同情况，区别不同的影响因素，广泛、灵活地运用多种多样的现代化管理办法来解决当代质量问题。目前，质量管理中有统计方法和非统计方法之分，常用的质量管理方法有所谓的老 7 种工具和新 7 种工具，其中老七种工具是指因果图、排列图、直方图、控制图、散布图、分层图和调查表，新 7 种工具是指关联图法、KJ（创始人日本川喜田二郎姓名的英文 Kawaklda Jiro 的缩写）法、系统图法、矩阵图法、矩阵数据分析法、过程决策程序图法（Process Decision Program Chart，PDPC）法和矢线图法。另外，还有一些新方法近年来得到了广泛的关注，例如，质量功能展开、田口方法、故障模式和影响分析、头脑风暴法、六西格玛法、水平对比法等。

上述“三全一多样”，都是围绕着“有效地利用人力、物力、财力、信息等资源，以最经济的手段生产出用户满意的产品”这一组织目标的，这是 TQM 的基本要求。

希赛教育专家提示：根据 TQM 的定义，可以把 TQM 看成是一种系统化、综合化的管理方法或思路，组织要实施 TQM，除了注意满足“三全一多样”的要求外，还必须遵循一定的原则并且按照一定的工作程序运作。

2．实施全面质量管理应遵循的原则

（1）领导重视并参与。组织领导应对组织的产品（服务）质量负完全责任，因此，质量决策和质量管理应是组织领导的重要职责。国内外实践已证明，开展 TQM，组织领导首先必须在思想上重视，强化自身的质量意识，带头学习、理解 TQM 并亲身参与 TQM。

这样，才能对组织开展 TQM 形成强有力的支持，促进组织的 TQM 工作深入扎实、持久地开展下去。

（2）抓住思想、目标、体系、技术 4 个要领。TQM 是一种科学的管理思想，体现了与现代科学技术和现代生产相适应的现代管理思想。因此，在推行 TQM 过程中，必须在思想上树立起市场的观念、竞争的观念、以用户为中心的观念，以及不断改进质量等其他一系列适应市场经济和知识经济时代的新观念。在此基础上，不断强化质量意识，综合地、系统地不断改进产品和服务的质量，持续满足用户的要求。

TQM 必须围绕一定的质量目标来进行。通过明确的目标，引导组织各方面的活动，激发全体职工的积极性和创造性，进而衡量和监控各方面质量活动的绩效。只有确立明确的质量目标，才有可能针对这个目标综合地、系统地推进全面质量管理工作。

组织的质量目标是通过一个健全而有效的体系来实现的。质量管理的核心是质量管理体系的建立和运行。通过建立和运行质量管理体系可以使影响产品和服务质量的所有因素，包括人、财、物、管理等，以及所有环节，涉及到组织中所有部门和人员，都处于控制状态，在此基础上，就可以确保质量目标的实现。其次，通过建立和运行质量管理体系，可以使所有部门围绕质量目标形成一个网络系统，相互协调地为实现质量目标努力。

TQM 是一套能够控制质量，提高质量的管理技术和科学技术。它要求综合、灵活地运用各种有效的管理方法和手段，从而有效地利用组织资源，开发出满足用户需要的产品。

（3）切实做好各项基础工作。TQM 是全过程的质量管理，是从市场调研一直到售后服务的系统的管理。TQM 要切实取得实效，必须首先做好各项基础工作。基础工作搞好了，TQM 就能收到事半功倍的效果，就有利于取得成效。反之，基础工作搞得不好，不管表面工作如何有声有色，如同建立在沙洲上的大厦，随时都有坍塌的危险。

（4）做好各方面的组织协调工作。开展 TQM，必须进行组织协调，综合治理。首先必须明确各部门的质量职能，并建立健全严格的质量责任制，明确各有关部门在质量管理方面的职能并规定其职责，以及围绕一定的质量目标所承担的具体工作任务。此外，还必须建立一个综合性的质量管理机构，从总体上协调和控制上述各方面的职能。

质量管理体系开始运行之后，还要通过一系列的工作对质量管理体系进行监控，保证使之按照规定的目标持续、稳定地运行。这方面的工作包括质量成本的分析、报告，质量管理体系审核，以及对用户满意程度的调查等。宏观的质量认证制度、质量监督制度也是促进组织 TQM 工作的有效手段。

（5）讲求经济效益，把技术和经济统一起来。提高质量能带来组织和全社会的经济效益。在组织中推行 TQM，能够减少整个生产过程及各个工序的无效劳动和材料消耗，降低生产成本，生产出用户满意的产品，增强组织竞争能力，实现优质、高产、低耗、盈利，提高组织的经济效益，促进组织发展壮大。从宏观的角度讲，这又可以节约资源，

减少浪费，增加社会财富，为全社会带来效益。

3．实施全面质量管理的五步法

在具体实施 TQM 时，可以遵循五步法进行。这五步分别是：决策、准备、开始、扩展和综合。

（1）决策。这是一个决定做还是不做决策的过程。对于很多组织来说，由于存在各种各样的驱动力，因此他们有实施 TQM 的愿望，常见的动因有：组织有成为世界级组织的远景构想；组织希望能够保持领导地位和满足用户需求；也有的组织是由于面临不利的局面，如用户不满意、丢失了市场份额、竞争的压力、成本的压力等。TQM 的实施能够帮助组织摆脱困境，解决问题，因此，全 TQM 越来越受到世界范围内组织的关注。当然，为了能够做出正确的决策，组织的高层领导者必须全面评估组织的质量状况，了解所有可能的解决问题的方案，在此基础上进行决策：是否实施 TQM。

（2）准备。一旦做出决策后，组织就应该开始准备。第一，高层管理者需要学习和研究 TQM，对于质量和质量管理形成正确的认识。第二，建立组织，具体包括：组成质量委员会，任命质量主管和成员，培训选中的管理者。第三，确立远景构想和质量目标，并制订为实现质量目标所必需的长期计划和短期计划。第四，选择合适的项目，成立团队，准备作为试点开始实施全面质量管理。

（3）开始。这是具体的实施阶段。在这一阶段，需要进行项目的试点，在试点中逐渐总结经验教训。根据试点中总结的经验，来着手评估试点单位的质量状况，主要从 4 个方面进行：用户忠诚度、不良质量成本、质量管理体系以及质量文化。在评价的基础上发现问题和改进机会，然后进行有针对性的改进，包括人力资源、信息等。

（4）扩展。在试点取得成功的情况下，组织就可以向所有部门和团队扩展。首先，每个重要的部门和领域都应该设立质量委员会、确定改进项目并建立相应的过程团队。第二，还要对团队运作的情况进行评估。为了确保团队工作的效果，应该对团队成员进行培训，还要为团队建设以及团队运作等方面提供指导。第三，管理层还需要对于每个团队的工作情况进行全面的测评，从而确认所取得的效果。扩展过程需要有一定的时间，这项活动的顺利进行，要求高层领导强有力的领导和全员的参与。

（5）综合。在经过试点和扩展之后，组织就基本具备了实施全面质量管理的能力。为此，需要对于整个质量管理体系进行综合。通常需要从目标、人员、关键业务流程以及评审和审核这 4 个方面进行整合和计划。

- 目标。组织需要建立各个层次的完整的目标体系，包括战略（这是实现目标的总体规）、部门的目标、跨职能团队的目标以及个人的目标。
- 人员。组织应该对于所有的人员进行培训，并且授权给他们让其进行自我控制和自我管理，同时要鼓励团队协作。

- 关键业务流程。组织需要明确主要的成功因素，在成功因素基础上确定关键业务流程。通常来讲，每个组织都有 4 或 5 个关键业务流程，这些流程往往会涉及到几个部门。为了确保这些流程的顺畅运作和不断完善，应该建立团队负责每个关键业务流程，并且要指派负责人。团队运作的情况也应该进行测评。
- 评审和审核。除了对于团队和流程的运作情况进行测评外，组织还需要对于整个组织的质量管理状况进行定期的审核，从而明确组织在市场竞争中的地位，及时发现问题，寻找改进机会。在评审时通常要关注 4 个方面，分别是市场地位、不良质量成本、质量管理体系和质量文化。

11.5.7 六西格玛管理

希腊字母 σ（音 SIGMA，大写为 Σ）是统计学里的一个单位，表示与平均值的标准偏差。六西格玛质量水平表示在生产或服务过程中有 3.4 个缺陷/百万次，即达到 99.9997% 合格率。

1．六西格玛管理的概念

可以把六西格玛管理定义为：“获得和保持组织在经营上的成功并将其经营业绩最大化的综合管理体系和发展战略。是使组织获得快速增长的经营方式”。六西格玛管理是“寻求同时增加用户满意和组织经济增长的经营战略途径”，是使组织获得快速增长和竞争力的经营方式。它不是单纯的技术方法的引用，而是全新的管理模式。

六西格玛管理具有以下特点：

（1）比以往更广泛的业绩改进视角，强调从用户的关键要求以及组织经营战略焦点出发，寻求业绩突破的机会，为用户和组织创造更大的价值。

（2）强调对业绩和过程的度量，通过度量，提出挑战性的目标和水平对比的平台。

（3）提供了业绩改进方法。针对不同的目的与应用领域，这种专业化的改进过程包括：六西格玛产品/服务过程改进 DMAIC 流程，六西格玛设计 DFSS 流程等。

（4）在实施上由勇士（Champion）、大黑带（MBB）、黑带（BB）、绿带（GB）等经过培训、职责明确的人员作为组织保障。

（5）通过确定和实施六西格玛项目，完成过程改进项目。每一个项目的完成时间在 36 个月。

（6）明确规定成功的标准及度量方法，以及对项目完成人员的奖励。

（7）组织文化的变革是其重要的组成部分。

六西格玛使用 DPMO（100 万个机会里面，出现缺陷的机会）来计算，其计算公式如下：

$$DPMO = 总的缺陷数/总机会数$$

根据以上公式，六西格玛各个级别的划分大致如表 11-3 所示。

表 11-3　六西格玛各级别的划分

一西格玛	二西格玛	三西格玛	四西格玛	五西格玛	六西格玛
690 000	308 000	66 800	6 210	230	3.4
说明：上面数字的单位为个缺陷数/百万机会					

使用六西格玛管理的公司努力的最终目标就是达到“六”，它代表最高境界的完美水平。

2．六西格玛管理的理念

尽管没有人对六西格玛管理理念和准则做出全面的概括，但是在下述方面，六西格玛管理较之其他管理模式和方法来说，更有侧重和强调。

（1）以用户为关注焦点。这不但是六西格玛管理的基本原则，也是现代管理理论和实践的基本原则。六西格玛管理强调倾听用户的声音。可以说，六西格玛管理是从倾听用户的声音开始的。在这方面人们往往受思维定式的束缚，总是认为“我们是这个领域的行家，我们还不知道用户需求吗？”。但是，当系统分析师将视角移到用户的角度，就会发现用户需求与自己认为的不尽相同。

六西格玛管理强调从了解用户开始，从确定用户的关键要求开始。很多实施六西格玛管理的组织，都将通过用户调查建立“用户仪表板（Customer Dashboard）”作为构建六西格玛管理基础的重要活动。正像“仪表板”这个词所表征的那样，“用户需求”要具体到关键的可测量的“指针”上——没有测量就没有管理，对用户满意程度亦是如此。

六西格玛管理正是在这种“用户需求”拉动下，从外向内分解和回溯至组织内部的关键要素、关键环节、关键流程、关键活动的。组织内部每一个六西格玛项目，都与“用户仪表板”相连，从而有效地支持用户满意程度的改进。

（2）系统观点。系统是由一系列相互关联、相互依赖、共同作用的过程构成的。在六西格玛管理中十分强调将组织作为系统来看待，而不是一些独立的部门和孤立的过程的集合。质量管理大师戴明博士在著名的“十四项法则”中指出：“85%以上的质量问题和浪费是由系统原因造成的，只有 15%是由岗位上的问题造成的”。

由于管理上的需要，组织在其内部需设立一系列职能部门，以负责完成具体的组织职能。但是，组织向用户提供产品和服务的活动是横向的，例如，由市场部签订合同，由设计部门设计，由生产部门制造，再由服务部门提供售后服务等，是由一系列环环相扣的过程构成的系统。应该说，组织的纵向结构对实现有序管理是必不可少的，但同时也带来了管理上的障碍。在很多组织内部，这些纵向结构如同一道道的墙。墙内的人考虑问题时往往更看重自己本部门的利益，而把问题“扔到墙外去”，使横向系统失去了协调。实际上，很多问题之所以长期得不到很好的解决，并不是组织不具备技术上的能力和条件，而是管理上的不协调。

（3）依据数据决策。“用数据说话”是六西格玛管理理念和原则的一个突出的特点，数据是测量的结果，也是分析和决策的依据。数据是过程运行情况的客观反映，数据以

它特有的方式告诉我们过程发生了什么以及改进的机会在何处。但遗憾的是，在很多组织中，“依据数据决策”还没有成为大家共同接受的管理原则，解决问题不是靠科学方法，而是靠个人智慧甚至是靠运气。

在六西格玛管理中，要求使用数据做出正确的统计推断，用数据帮助管理者准确地找到产生问题的根本原因。实现“依据数据做出决策”不但需要数据，还要有从数据得出信息的技术。统计技术正是这样一种技术。因此，在六西格玛管理中大量地用到统计技术。当然，六西格玛管理也绝不仅仅是统计技术的应用。它仅是帮助我们从数据得出信息而已，六西格玛管理涵盖的内容，不论从广度还是从深度上来说，比统计技术要多得多。

（4）关注过程管理。通过过程的优化实现组织竞争力的提高是六西格玛管理的核心理念。一个有竞争力的组织应该具备以“最高的质量、最快的速度、最低的价格”向用户或市场提供产品或服务的能力，而这个能力取决于组织核心业务过程的过程能力。在六西格玛管理中，这个能力表征为过程输出的结果与用户要求的一致性。

关于六西格玛管理理念和原则还有很多。例如，管理系统整合的理念。六西格玛是从全面质量管理的理论和最优实践中发展而来的，成功的六西格玛管理将所有有效的业绩改进发放整合在一起。

11.6 人力资源与沟通管理

项目人力资源管理就是管理好项目开发团队，提高工作绩效。项目经理有 80%左右的时间用在沟通上，包括内部沟通和外部沟通。因此，就项目经理本人而言，在人力资源管理和沟通管理上的技能，是所有技能中最重要的。

11.6.1 项目组织与项目经理

组织结构一般分为三种类型：职能型、项目型和矩阵型，其中矩阵型组织一般又分为三种：弱矩阵、平衡矩阵、强矩阵，如表 11-4 所示。项目经理在不同类型的组织中所拥有的权力和扮演的角色是不同的，同时不同的组织结构也制约着项目获得资源的能力。

表 11-4 组织结构对项目的影响

组织类型／项目特点	职能型组织	矩阵型组织			项目型组织
		弱矩阵	平衡矩阵	强矩阵	
项目经理的权威	很少或者没有	有限	中等	较大	很大
全职人员百分比	几乎没有	0%～25%	13%～60%	50%～95%	85%～100%
项目经理的职位	兼职	兼职	兼职	全职	全职
项目经理的头衔	项目协调人	项目领导人	项目经理	项目经理	项目经理
项目管理行政人员	兼职	兼职	兼职	全职	全职

不同的组织类型对项目的影响不同，各种组织类型对资源配置的方式，不同组织类型的优缺点和其所适应的竞争环境和市场环境。职能型组织有利于处理重复、比较固定的工作，而项目型组织比较适合创新、开发类的工作。项目型组织适合比较复杂、竞争激烈的市场环境。不同组织类型对人才培养也产生很大影响，职能型组织有利于培养全能型人才，项目型组织有利于培养专业技术型人才。矩阵型组织同时具有职能型组织和项目型组织的优点，是现在大部分公司所采用的组织形式。

信息技术项目成功最重要的三个方面是领导层的支持、用户的参与和明确的需求说明。要取得项目的成功，项目经理应该具有以下 4 个方面的能力：

（1）管理能力：包括一般的管理技能，例如，计划、控制、激励等，为了应对项目复杂的情况，必须有超强的解决问题能力。

（2）领导能力：为项目确定正确的方向，统一项目组成员的思想，并不断鼓舞和激励每一个人。

（3）沟通能力：能够和项目关系人产生良好的关系，适时并正确的和不同的关系人沟通，协调他们之间的关系。

（4）政治技巧：这里主要强调和领导层及其组织中其他部门沟通，提高项目获取资源的能力。

（5）专业技能：和项目相关的技能、技术，这对与 IT 项目也是非常重要的。

希赛教育专家提示：项目经理必须了解管理学的一些基本的理论，理解项目管理所需要技能的含义，掌握解决 IT 项目管理中最常产生问题所需要的技能。

11.6.2 项目人员管理

项目人力资源管理也可以理解为对人力资源的取得、培训、保持和利用等方面所进行的计划、组织、指挥和控制活动。具体而言，包括以下内容：

（1）人力资源计划。是指项目为了实现其目标而对所需人力资源进行预测，并为满足这些需要而预先进行系统安排的过程。

（2）工作分析。是收集、分析和整理关于某种特定工作的信息的一个系统性程序，工作分析要具体说明为成功地完成该项工作，每一个人的工作内容、必要的工作条件和员工的资格是什么。工作分析信息被用来计划和协调几乎所有的人力资源管理活动。

（3）员工招聘。根据项目任务的需要，为实际或潜在的职位空缺找到合适的候选人。

（4）员工培训和开发。指为了使员工获得或改进与工作有关的知识、技能、动机、态度和行为，以利于提高员工的绩效以及员工对项目目标的贡献，组织所作的有计划、有系统的各种努力。培训聚焦于目前的工作，而开发则是为员工准备可能的未来工作。

（5）报酬管理。通过建立公平合理的薪水系统和福利制度以起到吸引、保持和激励员工很好地完成其工作的作用。

（6）绩效评估。是对工作行为的测量过程，即用过去制定的标准来比较工作绩效的

记录以及将绩效评估的结果反馈给员工的过程。

以上是项目人力资源管理的核心内容，它们从管理程序上来讲，已经在很大程度上规范化了，从管理部门上来讲，也拥有专门的人力资源管理部门，因此属于制度化的人力资源管理。除了有章可循，程序比较固定的这一部分外，还有一些无固定做法可言的内容，这一部分一般为非组织化的人力资源管理，或更高层次的人力资源管理，主要包括领导艺术、群体激励、管理沟通、企业文化建设等内容。

11.6.3 IT 项目中的沟通

很多项目的失败跟沟通有很大关系，所以，在此先抛出一个论断：在成功的项目中很难衡量沟通起的作用，但失败的项目无一例外存在沟通问题。也就是说，沟通在项目中起的作用很微妙，往往是潜移默化的，是项目管理中的一个软性指标。如果一个项目做得很成功，很难分辨出合作各方的友好沟通到底起了多大作用；可是一旦项目失败，在总结教训的时候，总会提及沟通不畅的问题。

项目沟通管理是现代项目管理知识体系中的九大知识领域之一。项目沟通管理为成功所必需的因素——人、想法和信息之间提供了一个关键连接。涉及项目的任何人都应准备以项目“语言”发送和接收信息，并且必须理解他们以个人身份参与的沟通怎样影响整个项目。

1．IT 项目中沟通的特殊意义

对于项目来说，要科学地组织、指挥、协调和控制项目的实施过程，就必须进行信息沟通。沟通对项目的影响往往是潜移默化的。所以，在成功的项目中，人们往往感受不到沟通所起的重要作用，在失败项目的痛苦反思中，却最能看出沟通不畅的危害。没有良好的信息沟通，对项目的发展和人际关系的改善，都会起到制约作用。沟通失败是 IT 项目求生路上最大的拦路虎。常常能听到的典型例子是某某集团耗资几千万的 ERP 项目最终弃之不用，原因是开发出的软件不是用户所需要的，不但没有提高用户的工作效率，反而增加了工作量，而造成这种尴尬局面的根本原因是沟通失败。当一个项目组付出极大的努力，而所做的工作却得不到客户的认可时，是否应该冷静地反思一下双方之间的沟通问题？软件项目开发中最普遍的现象是一遍一遍地返工，导致项目的成本一再加大，工期一再拖延，为什么不能一次把事情做好？原因还是沟通不到位。

通常的项目管理教材将项目沟通的重要性归结为以下 4 点：

（1）决策和计划的基础。项目团队要想做出正确的决策，必须以准确、完整、及时的信息作为基础。通过项目内、外部环境之间的信息沟通，就可以获得众多的变化的信息，从而为决策提供依据。

（2）组织和控制管理过程的依据和手段。在项目团队内部，没有好的信息沟通，情况不明，就无法实施科学的管理。只有通过信息沟通，掌握项目团队内的各方面情况，才能为科学管理提供依据，才能有效地提高项目团队的组织效能。

（3）建立和改善人际关系是必不可少的条件。信息沟通、意见交流，将许多独立的个人、团体、组织贯通起来，成为一个整体。信息沟通是人的一种重要的心理需要，是人们用以表达思想、感情与态度，寻求同情与友谊的重要手段。畅通的信息沟通，可以减少人与人的冲突，改善人与人、人与团队之间的关系。

（4）项目经理成功领导的重要手段。项目经理通过各种途径将意图传递给下级人员并使下级人员理解和执行。如果沟通不畅，下级人员就不能正确理解和执行领导意图，项目就不能按经理的意图进行，最终导致项目混乱甚至项目失败。因此，提高项目经理的沟通能力，与领导过程的成功与否关系极大。

除以上 4 点外，结合 IT 项目的特点，IT 项目沟通的重要性还包括下面两点：

（1）信息系统本身是沟通的产物。软件开发过程实际上就是将手工作业转化成计算机程序的过程。不像普通的生产加工那样有具体的有形的原料和产品，软件开发的原料和产品就是信息，中间过程中传递的也是信息，而信息的产生、收集、传播、保存正是沟通管理的内容。可见，沟通不仅仅是软件项目管理的必要手段，更重要的，沟通是软件生产的手段和生产过程中必不可少的工序。

（2）软件开发的柔性标准需要沟通来弥补。软件开发不像加工螺钉、螺母，有很具体的标准和检验方法。软件的标准柔性很大，往往在用户的心里，好用、满意是软件成功的标准。而由于用户的能力所限，这个标准在软件开发前很难确切地完整地表达出来。因此，开发过程中项目组和用户的沟通互动是解决这一现实问题的唯一办法。

希赛教育专家提示：在 IT 行业，沟通的成败决定整个项目的成败，沟通的效率影响整个项目的成本、进度，沟通不畅的风险是 IT 项目的最大风险之一。

2. IT 项目中沟通的误区

在这个小节中，笔者就自己的一些实践经验，谈谈 IT 项目中沟通的一些误区。

（1）试图用技术的手段来解决只有通过沟通才能解决的问题。笔者本人有过多年的研发团队管理经验，发现技术人员有一个普遍的缺点，那就是相对而言不太重视沟通。很多技术人员特别喜欢和产品较劲，和硬件较劲，而不善于和客户沟通，对客户提出的需求不管合理与否一般都不会说不，一概答应。然后按照自己的理解，加班加点做二次开发来满足客户的需求，遗憾的是，当他费尽九牛二虎之力完成开发之后拿给客户看，本以为客户会大加赞赏，却听到客户说：“我要的不是你做的这样，你曲解了我的意思。”结果技术人员两三个星期加班的工作成果得不到认可，造成很大的劳动力资源浪费，而因为研发人员的不善沟通发生的此类事件还在不断上演。

从中不难分析出问题的症结在于有很多问题是需要和客户充分沟通，共同探讨解决方案的。比如有些业务可以通过客户方业务流程的稍微调整实现，而技术人员却通常因为拙于沟通，觉得说服客户业务流程调整需要较强沟通能力，于是就试图通过自己擅长的技术手段来解决问题，比如做二次开发。很多技术人员不善于沟通，甚至没有欲望把自己的观点表达出来，不善于和客户统一思想，就只能一味迁就客户，全盘接收客户恰

当或不恰当、合理或不合理的需求，回过头来，只能期望用技术的手段加班加点来解决。遗憾的是，有些问题只能通过沟通来解决，沟通到位了，便会事半功倍，根本不需要投入巨大工作量。

（2）采用回避沟通的鸵鸟战术。有时候当项目进展不理想，实施中存在问题或矛盾时，实施顾问往往因为担心受到批评和埋怨，不敢或不愿意汇报给上级，把问题和困难藏着、掖着，却不知已经失去了解决问题或困难的最佳时机。然而，藏着、掖着并不能解决问题，只能拖延，最后的结果可能是矛盾越积越大，问题越积累越复杂，直到实施顾问实在兜不住的时候再爆发出来，那时候的局面就很难收拾。这也恰好印证了墨菲定律：越怕爆发的事情越会爆发。所以，千万不能隐瞒不好的消息，否则很可能失去了解决问题的最佳时机。

还有一点就是想当然、满怀假设。因为不善于沟通，实施顾问就想当然地假设客户的需求，用自己的假设来代替没有调研清楚的客户需求，想当然地认为客户的需求就是自己想的那样，而事实上，客户的真正需求和实施顾问的假设往往相反。很多人在实施方案上遇到分歧，不去问客户到底应该怎么处理，而是假设自己是客户，用自己的意愿选择一条路，结果沿着这条路走了很远以后，才发现客户偏偏想走的是另一条路。所以说，这种以假设为依据的决策，往往因为假设是错的，结论也是错的。这种想当然的假设造成沟通障碍，势必引发诸多误会，造成很多返工。

（3）忽视情感上的沟通。我觉得在项目实施过程中，情感上的沟通非常重要。因为归根结底，项目实施过程是顾问方向客户方提供的一种服务，凡是服务，对应到客户那边就是一种体验。既然是体验，就是要给客户一种感觉，这些感觉中既有理性的成分也有感性的成分，而客户的认知往往是感性的因素居多。比如说，软件移用性的问题，操作的难易固然有客观的、理性的标准，但受感性因素的影响很大，再难操作的东西，用心学，学会使用也就不难操作了，所以，只要下决心学习，操作就不难，如果没有兴趣学习，操作就非常难，从这个意义上讲，所谓的易用性受情感的影响很大，有非常感性的、柔性的标准。这就要求实施顾问在和客户沟通的时候，不仅仅要谈工作话题，也不能忽略了情感上的沟通。实际上，情感和氛围对项目的效率乃至成败影响很大。

（4）陷入无休止的争论。还有一种情况必须警惕并尽量避免，那就是沟通目标不明确，偏离主题，甚至陷入无休止的争论。这种现象好像普遍存在于各种组织中，试问，全国每年有多少无休止的争论和无效率的会议在进行？这些争论所消耗的生产力是惊人的！什么叫无休止的争论？可以举个例子：

A 和 B 之间沟通。车间的领导对 A 职员说：“小 A 呀，你去库房给咱们车间领 10 个螺母”。A 就跑去领料，库管 B 女士说：“不行，你必须办理特殊申请手续，不然这个料不让领”。A 说：“凭什么呀？车间紧急用，你一个库管卡这点小事情，耽误生产你担得起责任吗”？于是，两个人为了这个手续就较上劲了。A 说 B 一个破库管有什么了不起的，B 说全厂就你特殊，有什么了不起的。于是两个人争执的焦点无形中就从领螺母

转移到比谁更了不起上来，各自开始兜家底，一个滔滔不绝说道：我家大爷是干什么的，我家叔叔是干什么的，我爱人是干什么的，我就是比你了不起。结果，另外一个人当然也毫不示弱，也比划一番。最后，两人都把领料这件事给忘了，演变成比家庭和社会关系了，这样的话，沟通就成了一个无休止的争论、无意义的争论。

这个例子在现实生活中时有发生，大家都觉得很滑稽，但实际上，我们有很多会议就是这样进行的，开来开去不经意间就转移了主题，本来为了解决问题的会，最后演变成争论企业文化和激励制度。还有一些无休止的争论，是与会双方为了维系自己的面子或推卸责任而争论，也是经常会发生的。

所以，不是所有的人坐在一起这种沟通就有效，而是要有目的地沟通，是就事论事地沟通，沟通之前就要把为什么来沟通，沟通要达到什么目的，非常明确地界定出来，而不是随便坐坐，东扯西扯，闲聊唠嗑，所以沟通一定要有目标地就事论事地沟通。

3．通过改善沟通来提高效率

事实上，沟通对项目推进效率影响很大，良好的团队氛围和沟通环境能够大大提高项目实施效率，IT 项目成败的决定因素是人，如果团队中的人均效率能够提高 1%或人均成本能够降低 1%的话，那么，整个项目的效率可能就会提高好几倍！团队越大越是如此。所以说，默契是项目组最宝贵的财富，团队的默契程度对软件项目的实施效率影响很大。一个经过长期磨合、相互信任、形成一套默契的做事方法和风格的团队，可以省掉很多不必要的沟通。相反，初次合作的团队因为团队成员各自的背景和风格不同、成员间相互信任度不高等原因，要充分考虑沟通消耗。

软件企业人员流动率高的特点导致团队凝聚力和默契度的培养比较困难。而凝聚力和默契度是需要长期的、大量的内部沟通和交流才能逐步形成，由此不难理解持续良好的沟通和交流是一个团队的无形资产，稳定、默契的开发团队是一个软件企业的核心竞争力的道理。

还有一点不容忽视，那就是软件开发这种以人脑为主要工具的创造性很强的作业，开发人员的心情和兴奋度对个人工作效率影响很大，而一个人置身于氛围良好、合作默契的团队中心情一般较好，这种良好的氛围所能带来的能量是不可估量的。

11.7　项目风险管理

项目是在复杂多变的环境中进行的，整个过程受诸多因素的影响，项目团队对这些影响项目正常进展的内外因素往往很难判断和加以控制。项目同其他经济活动一样带有风险。要避免和减少损失，将威胁化为机会，项目团队就必须了解和掌握项目风险的来源、性质和发生规律，进而实行有效的管理。项目风险管理是指识别、分析并对项目风险做出积极反应的系统过程。通过主动、系统地对项目风险进行全过程识别、评估及监控，达到降低项目风险、减少风险损失，甚至化险为夷，变不利为有利的目的，包括项

目风险管理计划、风险识别与分析、监控和应对过程。

11.7.1 风险管理计划

风险管理计划过程包括定义项目组织及成员风险管理的行动方案及方式，选择合适的风险管理方法，确定风险判断的依据等，风险计划就是制定风险规避策略以及具体实施措施和手段的过程。风险管理计划的依据如下：

（1）项目整体计划的相关内容，如项目目标、项目规模、项目利益相关者情况、项目复杂程度、所需资源、项目时间段、约束条件及假设前提等可作为计划的依据。

（2）项目组织及个人所经历和积累的风险管理经验及实践。

（3）决策者，责任方及授权情况。

（4）项目利益相关者对项目风险的敏感程度及可承受能力。

（5）可获取的数据及管理系统情况。丰富的数据和严密的系统基础，将有助于风险识别、评估、定量化及对应策略的制定。

（6）风险管理模板。项目经理及项目组将利用风险管理模板对项目进行管理，从而使风险管理标准化、程序化。模板应在管理的应用中得到不断改进。

风险管理计划一般通过计划会议的形式制定，会议参加人员应包括项目经理，团队领导者及任何与风险管理计划和实施相关者，计划会议将具体的把风险管理标准模板应用于当前的项目。

风险管理计划应包括以下内容。

（1）方法：确定风险管理使用的方法、工具和数据资源，这些内容可随项目阶段及风险评估情况做适当的调整。

（2）人员：明确风险管理活动中领导者、支持者及参与者的角色定位、任务分工及其各自的责任。

（3）时间周期。界定项目生命周期中风险管理过程的各运行阶段，及过程评价、控制和变更的周期或频率。

（4）类型级别及说明。定义并说明风险评估和风险量化的类型级别。明确的定义和说明对于防止决策滞后和保证过程连续是很重要的。

（5）基准。明确定义由谁以何种方式采取风险应对行动。合理的定义可作为基准衡量项目团队实施风险应对计划的有效性，并避免发生项目业主方与项目承担方对该内容理解的二义性。

（6）汇报形式。规定风险管理各过程中应汇报或沟通的内容、范围、渠道及方式。汇报与沟通应包括项目团队内部之间的，及项目外部与投资方及其他项目利益相关者之间的。

（7）跟踪。规定如何以文档的方式记录项目过程中风险及风险管理的过程，风险管理文档可有效用于对当前项目的管理，项目的监察，经验教训的总结及日后项目的指导。

11.7.2　风险识别

风险识别就是确定风险的来源，研究风险事件是否会影响到本项目。风险识别实际上是一种预测，是对项目未来情况的设想。由于风险的不确定性，不可能一次就把所有的风险都识别出来，需要在项目自始至终的过程中定期进行。

1. 风险识别的依据

风险识别的基本依据是客观世界的因果关联性和可认识性。具体进行风险识别时，要充分考虑：

（1）项目成果说明。项目成果的性质决定了项目会有哪些风险。使用成熟技术的成果在所有其他因素相同的情况下遇到的风险比需要创新和发明的成果要少。

（2）制约因素和假定。项目专项计划，例如，范围计划、质量计划、成本计划、进度计划等，都是在某些制约因素和假设下进行的，对这些计划详细审查，可能找出其中预示机会或威胁的因素或假定。

2. 风险识别的手段

识别风险时可利用的手段有以下一些。

（1）风险检查表：有经验的项目组会根据经验罗列出可能发生的风险，作为风险检查表，能够提醒人们可能发生的风险。

（2）以往的结果：实际进行的或以前的项目结果对于识别潜在的机会和风险极有帮助。

（3）问卷调查：问卷调查可能发生的风险也是很好的办法。

（4）事件树：事件树分析法就是从项目起始状态出发，根据项目工作分解结构，通过逻辑推理，确定风险事件发生、发展和演变的过程。

3. 风险识别的结果

风险识别结束时应当把风险来源和潜在的风险事件进行罗列和分类，并说明如下事项。

（1）风险来源：对项目有积极或消极影响的风险事件。例如，不明确的成本估算、项目团队人员的轮换更迭、不完善的合同文件、难以琢磨的合作伙伴等。

（2）风险症状：风险症状就是风险事件的各种间接表现，包括苗头和征兆。

（3）对其他过程的要求：通过风险识别，有可能发现有必要在项目其他方面进行改进。如发现工作结构分解做得不够详细，不能够充分识别各种风险。

11.7.3　风险分析与量化

风险量化的基本目的是确定哪些事件需要制定应对措施。对于同一风险，不同的组织和个人的承受能力会有不同，所以风险量化要以风险识别的结果和项目干系人对风险的承受能力为依据，哪些项目风险需要量化在很大程度上取决于项目干系人的风险承

受力。

在风险量化中，有些情况需要特别注意：

（1）机会和威胁在一定条件下可相互转化。

（2）风险事件可能造成多种后果。

（3）对项目某一干系人的机会，对另一干系人可能就是一种威胁。

风险是否需要制定应对措施，要看其大小。表示风险大小有多种方法。

（1）风险事件预期价值：风险事件预期价值是风险事件概率和风险后果价值的乘积。风险后果价值是对风险事件带来的收益或造成的损失量化估算。在估算风险后果价值时要反映有形的，也要反映无形的价值。

（2）统计和。就做为项目成本、工作量、时间以及其他事项的估算。

（3）进度模拟。进度模拟常用项目的网络图做为模型，项目管理软件可以实现项目的模拟实施，通过多次模拟实施测算,可以得到风险影响的统计分布。

（4）决策树。决策树是一种便于决策者理解的，来说明不同决策之间和相关偶发事件之间的相互作用的图表。决策树的分支或代表决策，或代表偶发事件，用事件发生的概率和所要付出的代价量化风险的大小。

11.7.4 风险应对

可从风险后果的性质、风险发生的概率或风险后果大小等维度来综合决策风险应对策略。常见的策略有减轻、预防、转移、回避、自留和后备等，具体采取哪一种（或几种），取决于项目的风险形势。

（1）减轻风险。此策略的目标是降低风险发生的可能性或减少后果的不利影响。指导思想就是将项目每一个具体“风险”都减轻到可接受的水平。具体的风险减轻了，项目整体失败的概率就会减小，成功的概率就会增加。根据帕累托二八原理，项目所有风险中只有一小部分对项目威胁最大。因此，要集中力量专攻威胁最大的那几个风险。

（2）预防风险。预防策略通常采取有形和无形的手段，对那些常见的风险提前采取预防措施。有些风险，可以使用成熟的预防策略，例如，外汇风险，如果项目的投入或产出涉及到外汇，则项目团队必须采取措施预防外汇风险。

（3）转移风险。转移风险又叫合伙分担风险，其目的不是降低风险发生的概率和不利后果的大小，而是借用合同或协议，在风险事故一旦发生时将损失的一部分转移到项目以外的第三方身上。

（4）回避风险。回避是指当项目风险潜在威胁发生可能性太大，不利后果也太严重，又无其他策略可用时，主动放弃项目或改变项目目标与行动方案，从而规避风险的一种策略。如果通过风险评价发现项目的实施将面临巨大的威胁，项目管理团队又没有别的办法控制风险，甚至保险公司亦认为风险太大，拒绝承保，这时就应当考虑放弃项目的实施，避免巨大的人员伤亡和财产损失。

（5）自愿接受。有些时候，项目团队可以把风险事件的不利后果自愿接受下来。自愿接受可以是主动的，也可以是被动的。由于在风险管理计划阶段已对一些风险有了准备，所以当风险事件发生时马上执行应急计划，这是主动接受。被动接受风险是指在风险事件造成的损失数额不大，不影响项目大局时，项目团队将损失列为项目的一种成本。

（6）后备措施。有些风险要求事先制定后备措施。一旦项目实际进展情况与计划不同，就动用后备措施。主要有成本、进度和技术三种后备措施。

本章参考文献

[1] 白思俊. 现代项目管理. 北京：机械工业出版社，2002

[2] 中国项目管理委员会. 中国项目管理知识体系与国际项目管理专业资质认证标准. 北京：机械工业出版社. 2001

[3] 吴之明、卢有杰. 项目管理引论. 北京：清华大学出版社，2000

[4] （美）项目管理协会. 项目管理知识体系指南. 第 3 版. 北京：电子工业出版社，2006

[5] 郑人杰，殷人昆，陶永雷. 实用软件工程. 北京：清华大学出版社，2001

[6] 马林，罗国英. 全面质量管理基本知识. 北京：中国经济出版社，2001

[7] 郝志安. 全面质量管理简介. http://www.csai.cn/SoftQuality/No024.htm

[8] 孙静. 六西格玛手册. 北京：清华大学出版社，2003

[9] 鲍海燕. 如何实施软件质量保证. http://www.ccw.com.cn/htm/work/hr/02_8_8_2.asp

[10] Roger S Pressman. 软件工程：实践者的研究方法. 第 4 版. 黄柏素，梅宏译. 北京：机械工业出版社，1999

第 12 章　企业信息化战略与实施

就一般意义而言，企业信息化的目的就是要建立一个整体上相当于人的神经系统的数字神经系统。这种数字神经系统，使得企业具有平稳和有效地运作的能力，对紧急情况和机会做出快速反应，为企业内外部用户提供有价值的信息，以提高企业的核心竞争力。

企业要应对全球化市场竞争的挑战，特别是大型企业要实现跨地区、跨行业、跨所有制、跨国经营的战略目标，要实施技术创新战略、管理创新战略和市场开拓战略，要将企业工作重点转向技术创新、管理创新和制度创新的方向上来，信息化是必然选择和必要手段。企业信息化涉及到对企业管理理念的创新，管理流程的优化，管理团队的重组和管理手段的革新。

12.1　企业信息化规划

所谓信息化，可以认为是现代信息技术与社会各个领域、各个层面相互作用的动态过程及其结果。在这一相互作用过程中，信息技术自身和整个社会都发生着质的变化。其中，社会的质的变化主要表现为信息资源开发和应用，以及知识生产力迅速提高的结果。信息化是与当代信息革命、信息社会相关联的，信息化不同于工业化，工业化是信息化的基础，信息化可以促进工业化的进程；信息化不等同于现代化，在现代的时代背景下，信息化是现代化的目标之一；信息化不等于自动化，传统的自动化设备是以物质能源来驱动的，而对于信息化设备而言，信息不仅是处理对象，而且是信息系统的资源。

从本质上看，信息化应该是以信息资源开发利用为核心，以网络技术、通信技术等为依托的一种新技术扩散的过程。作为这一过程的结果，它将最终会引起整个产业结构的变化。

12.1.1　信息化的内容

信息化是一个非常宽泛和宏观的概念，而当人们谈到信息化时，总是具体的和有针对性的。我国国家信息化管理部门列出了国家信息化体系的 6 个要素，可以作为区域信息化、行业信息化、企业信息化等的参考。

（1）信息资源。信息和材料、能源共同构成经济和社会发展的三大战略资源。我国信息资源极其丰富，但开发利用的程度较低，远远落后于需要。因此，开发和利用信息资源是我国信息化的关键一环和决定性的一环。

（2）信息网络。信息网络是信息资源开发、利用的基础设施，信息网络包括计算机网络、电信网、电视网等。信息网络在国家信息化的过程中将逐步实现三网融合，并最终做到三网合一。

（3）信息技术应用。信息技术应用是国家信息化中十分重要的要素，它直接反映了效率、效果和效益。

（4）信息产业。信息产业是信息化的物质基础。信息产业包括微电子、计算机、电信等产品和技术的开发、生产、销售，以及软件、信息系统开发和电子商务等。从根本上来说，国家信息化只有在产品和技术方面拥有雄厚的自主知识产权，才能提高综合国力。

（5）信息化人才。人才是信息化的成功之本，而合理的人才结构更是信息化人才的核心和关键。合理的信息化人才结构要求不仅要有各个层次的信息化技术人才，还要有精干的信息化管理人才、营销人才，法律、法规和情报人才。在信息化人才中有一种人才最为重要，那就是系统分析师。系统分析师既是信息化的技术人才，同时，又是经营管理人才，是一种复合型人才。而首席信息官（Chief Information Officer，CIO）又是系统分析师队伍的领军人物，是企业最高管理层的重要成员。

（6）信息化政策、法规、标准和规范。信息化政策和法规、标准、规范是国家信息化快速、有序、健康和持续发展的保障。

企业信息化是指企业以业务流程的优化和重构为基础，在一定的深度和广度上利用计算机技术、网络技术和数据库技术，控制和集成化管理企业生产经营活动中的各种信息，实现企业内外部信息的共享和有效利用，以提高企业的经济效益和市场竞争力。

如果从动态的角度来看，企业信息化就是企业应用信息技术及产品的过程，或者更确切地说，企业信息化是信息技术由局部到全局，由战术层次到战略层次向企业全面渗透，运用于流程管理、支持企业经营管理的过程。这个过程表明，信息技术在企业的应用，在空间上是一个由无到有、由点到面的过程；在时间上具有阶段性和渐进性，起初是战术阶段，经过逐步深化，发展到战略阶段；信息化的核心和本质是企业运用信息技术，进行隐含知识的挖掘和编码化（文档化），进行业务流程的管理。企业信息化的实施，一般来说，可以沿两个方向进行，一是自上而下，必须与企业的制度创新、组织创新和管理创新结合；二是自下而上，必须以作为企业主体的业务人员的直接受益和使用水平逐步提高为基础。

12.1.2 信息化规划的内容

企业信息化一定要建立在企业战略规划基础之上，以企业战略规划为基础建立的企业管理模式是建立企业战略数据模型的依据。

企业信息化就是技术和业务的融合。这个“融合”并不是简单地利用信息系统去对手工作业流程进行自动化，而是需要从三个层面来实现。

首先，企业战略的层面。在规划中必须对企业目前的业务策略和未来的发展方向做深入分析。通过分析，确定企业的战略对企业内外部供应链的相应管理模式，从中找出实现战略目标的关键要素，分析这些要素与信息技术之间的潜在关系，从而确定信息技术应用的驱动因素，达到战略上的融合。

其次，业务运作层面。针对企业所确定的业务战略，通过分析获得实现这些目标的关键业务驱动力和实现这些目标的关键流程。这些关键流程的分析和确定要根据他们对企业价值产生过程中的贡献程度来确定。关键的业务需求是从那些关键的业务流程的分析中获得的，它们将决定未来系统的主要功能征。这一环节非常重要，因为，信息系统如果能够与这些直接创造价值的关键业务流程相融合，这对信息化投资回报的贡献是非常巨大的，也是信息化建设的成败的一个衡量指标。

再次，管理运作层面。虽然这一层面从价值链的角度上来说，是属于辅助流程，但它对企业的日常管理的科学性、高效性是非常重要的。另外，在企业战略层面的分析中，可以获得适应企业未来业务发展的管理模式，这个模式的实现离不开信息技术的支撑。所以，在管理运作层面的规划上，除了提出应用功能的需求外，还必须给出相应的信息技术体系，这些将确保管理模式和组织架构适应信息化的需要。

企业信息化规划的重要性是不言而喻的，但是，要防止一种倾向，就是把信息化规划片面地理解为信息技术规划，这样的观念是有害的。

企业战略数据模型分为数据库模型和数据仓库模型，数据库模型用来描述日常事务处理中数据及其关系；数据仓库模型则描述企业高层管理决策者所需信息及其关系。在企业信息化过程中，数据库模型是基础，一个好的数据库模型应该客观地反映企业生产经营的内在联系。数据库是办公自动化、计算机辅助管理系统、开发与设计自动化、生产过程自动化、Intranet 的基础和环境。有关数据仓库模型的详细知识，请阅读本套丛书中的《系统分析师技术指南（2009 版）》的第 6.2 节。

信息技术和网络技术都在飞速发展，企业信息化是多种类、多层次信息系统建设、集成和应用的过程，因此，不是一蹴而就的事情，需要结合企业的实际，全面规划，分步实施。

企业的信息化建设，不是为了信息化而信息化，其最终目的是为了满足企业的业务需求。因此，企业在考虑信息化建设时，首先必须明确企业业务对信息化的需求到底是什么。

如果企业对自身的业务发展战略和业务结构没有一个清晰的认识，对企业业务的信息化需求没有宏观的把握，那么，它的信息化建设往往不仅不能促进企业业务，反而可能产生负面影响。

企业的信息化建设是一项长期而艰巨的任务，不可能在短时间内完成。因此，企业的信息化建设任务必然会分解成各项相对独立的信息化建设项目，在不同时期分别进行，从而建立多个信息系统。如果缺乏总体规划，非常容易出现应用系统林立的情况，各种

系统相互孤立，互不连通，形成各个信息孤岛。而要把这些信息孤岛整合起来，其难度超乎寻常，甚至造成返工，从头重建。

要解决这些关系到企业信息化建设的全局性问题，就必须从全局出发，在战略层次上对企业的信息化建设进行总体规划。

12.1.3　信息化规划与战略规划

在全球化和信息化的大背景下，企业生存竞争环境发生了根本变化。这种巨大变化，主要表现在科技进步和产品更新速度不断加快，企业间的竞争可以超越时空的限制，本来身处一地的企业之间的竞争，都有可能带有国际竞争的性质。在严酷的内外部环境中，企业为了谋取长远发展，获得竞争优势并掌握战略主动权，一个重要的战略基础便是企业信息化。企业信息化是企业在信息时代谋求生存和发展的基本条件和必备素质。

1．企业战略需要信息化战略的支持

随着信息技术的迅猛发展和普及，企业战略的实现已经离不开信息技术和信息系统。

美国学者迈克尔•波特（Poter）在《竞争战略》一书中指出，企业在一个竞争环境中，往往面临 5 种竞争：供应商的讨价还价能力、购买者的讨价还价能力、潜在竞争者进入的能力、替代品的替代能力、行业内竞争者现在的竞争能力。这就是著名的“五力模型”，五力模型将大量不同的因素汇集在一个简便的模型中，以此分析一个行业的基本竞争态势。一种可行战略的提出，首先应该包括确认并评价这 5 种力量，不同力量的特性和重要性因行业和企业的不同而变化。

五力模型虽然没有直接描述信息技术对企业的竞争战略的影响，但是，仍然可以说明信息系统的引进和使用，对于企业实现这些战略是紧密相关和非常重要的。随着电子商务在企业经营管理中的广泛应用，缩短了企业与消费者的距离，极大地提高了企业获取新技术、新工艺、新产品的能力，给消费者提供了更多的选择；同时，也给企业带来更多的商机，要求企业必须提高了解市场和把握市场的能力。企业必须迅速将消费者的需求变化及时反映到决策层，促进企业针对消费者需求进行研究与开发，及时改变和调整经营战略，不断向市场提供差别化的产品和服务，形成不易为竞争对手模仿的独特竞争优势。

波特列出了企业常用的基本竞争战略，包括总成本领先战略、差别化战略和目标集聚战略等，而企业信息化正是对企业竞争战略提供支持。

2．企业战略指导企业信息化战略

信息技术是实现企业价值和企业战略的有效工具，而离开了企业战略需要去规划信息系统，犹如建设海市蜃楼。因此，企业信息化规划的价值源泉在于企业价值的实现，在于企业的经营战略和实际业务需要。

不少企业规划企业信息化时，往往是有一项需要就上一个系统，这样，每一个信息

系统都规划得规模庞大、覆盖全局，但系统之间信息隔阂、数据分散、功能重叠，其结果必然是导致严重浪费。问题的根源就在于企业在进行信息化规划之前，缺乏从企业战略的角度整体地统一考虑。大量实践经验证明，把信息化战略与企业经营战略结合起来建立的信息系统，其效果与不相结合有着显著的不同。因为企业经营战略是对企业长远发展的全局性谋划，是企业组织的奋斗目标，信息化并没有独立目标，而是为企业的战略目标服务的，为企业提供新的参加市场竞争的手段，从而大大提高企业市场竞争力。

因此，在制定企业信息化规划时，首先的任务是要详细研究企业的战略目标、方针策略，据此制定出信息化的战略目标，确定关键因素、关键决策以及约束条件等重要内容。只有在全局思想的指导下规划企业信息化，才能高屋建瓴、势如破竹。因此，规划企业信息化时，要遵循以下几条原则：一是支持企业战略目标，超前于企业发展步伐；二是建立与企业主导业务和主要业务流程协调一致的体系结构；三是加强信息资源综合管理；四是统一规划和分步实施。

希赛教育专家提示：企业信息化战略接受企业战略的指导并不是说信息化战略并不那么重要，而是恰恰相反。从信息化战略和企业战略的关系角度分析，对于成功的企业来说，企业信息化规划和企业战略规划几乎是同等重要。

从企业战略的角度来看，首先应当明确企业的任务、使命以及长远的发展目标。其次，要定义企业与外部经营环境的关系，明确基本的市场竞争的策略和方法。例如，企业的资金从哪里获得？企业的客户是谁？合作伙伴是谁？另外，企业还应当在目标、原则和政策指导下，统一各种不同的意见，避免使得企业战略流于形式。

另一方面，由于信息化对企业战略的影响重大，所以必须十分重视企业信息系统的规划和建设。企业信息系统可以以企业战略为基础来规划建设，也可以和企业战略的某些重要部分整合起来，或者完全与企业战略合为一体。这要根据具体的业务内容和信息技术特点来确定，使信息系统与企业的战略相辅相成，相互促进。

总而言之，在进入信息时代的过程中，信息、信息技术和信息系统作为一种资源已不再仅仅起着支撑企业战略的作用，而是在某种程度上决定企业战略。

3. 保持信息化规划与战略规划的一致性

企业信息化建设的核心问题是保证企业信息化战略与企业战略的一致性。在企业信息化实践中，如何处理信息化战略与企业战略的关系是一个比较薄弱的环节。在很多企业，二者分别独立运行，缺乏协调和统一，从而导致企业战略缺乏信息系统的支撑，同时，信息系统由于缺少企业战略的指导，而致其无效或低效。事实上，这也是造成 IT 黑洞的主要原因之一。

信息化规划与战略规划的关系有以下三种情况：

（1）当企业处在信息化的初级阶段时，业务部门根据现有的业务流程，或管理需要，直接提出信息化需求，信息技术部门按照需求实施。例如，财务部门提出财务电算化的需求，运作部门提出库房管理的需求，信息技术部门则根据不同部门的需要分别独立实

施，这样，就形成了一个完全基于企业组织与业务流程的信息系统结构，其中的各个信息系统分别对应于特定部门或特定业务流程。

（2）当企业处在信息化管理的中级阶段，企业制定了整体的经营战略，业务部门则根据企业经营战略，对现有的业务流程和组织进行变革，然后由不同的业务部门分别提出信息化需求，由信息技术部门分别独立实施。这时候形成的信息系统结构是与优化后的组织及业务流程结构相对应的。

（3）当企业处在信息化管理的高级阶段，企业会根据整体的经营战略，通盘考虑各业务部门的信息化需求，制定整体的信息化战略，统一规划，分步实施，此时建立起来的信息系统结构，由企业信息化战略统一指导，并与优化后的组织及业务流程相适应。

如何才能确保企业经营战略与信息化战略的一致性以及企业组织及业务结构与企业信息系统结构的一致性，是企业信息化战略制定中一个至关重要的问题，也是其中的难点之一。必须认识到，保持二者的一致性，不可能是一个静态的结果，而是一个动态的过程，因此，不会是一劳永逸。在实际工作中，信息化规划与战略规划总是互相影响、互相促进。可能正是有了信息化才暴露了企业原有战略的缺陷和问题，因此就必须对原有战略进行调整或修改。而由于原有企业战略的修改变动，就要求信息化战略也必须做相应修改。

12.1.4　信息系统战略规划方法

信息系统战略规划方法主要有业务系统规划（Business Systems Planning，BSP）、关键成功因素法（Critical Success Factors，CSF）、战略目标集合转化法（Strategy Set Transformation，SST）等。

1．业务系统规划方法

BSP 方法是由 IBM 公司研制的指导企业信息系统规划的方法，虽然研制始于 20 世纪 70 年代，但其方法和思想至今仍有指导意义。它辅助企业信息系统规划，来满足其近期和长期的信息化需求。

实行 BSP 的前提是，在企业内有改善信息系统的要求，并且有为建设这一系统而建立总的战略需要。因此，BSP 的基本功能是服务于信息系统建设的长期目标。可以将 BSP 看成是一个转化过程，即将企业的战略目标转化成信息系统的战略目标，因此，了解企业的战略就成了 BSP 的重要内容之一。

一般来说，不同层次的管理活动有着不同特点的信息化需求，因此，有必要建立一个合理的框架，并据此来定义信息系统。首先，信息系统应强调对管理决策的支持。一般认为，在任一企业内同时存在着三个不同的层次，即战略计划层、管理控制层和操作控制层。战略计划层是决定组织目标、达到这些目标所需用的资源，以及获取、分配这些资源的策略的过程；通过管理控制层，管理者确认资源的获取及组织的目标是否有效地使用了这些资源；操作控制层保证有效率地完成具体的任务。

希赛教育专家提示：BSP 方法确立了信息系统建设的若干原则。方法本身是可以灵活运用的，即方法中的某些步骤和技巧可根据具体情况变化而做相应的调整，但它的基本原则不能违背，因为这些原则是 BSP 的灵魂。

1）BSP 的目标

BSP 的目标主要是提供信息系统规划，用以支持企业短期和长期的信息需要。其具体目标可归纳如下：

- 为管理者提供一种形式化的、客观的方法，明确建立信息系统的优先顺序，而不考虑部门的狭隘利益，并避免主观性。
- 为具有较长生命周期系统的建设和投资提供保障。由于系统是基于业务活动过程的，因此，不因机构变化而失效。
- 为了以最高效率支持企业目标，BSP 提供数据处理和资源管理。
- 增加负责人的信心，使其坚信高效的信息系统能够被实施。
- 通过提供信息系统对用户需求的快速相应，从而改善信息系统管理部门和用户之间关系。
- 应将数据作为一种企业资源加以确定，为使每个用户更有效地使用这些数据，要对些数据进行统一规划、管理和控制。
- 由 BSP 所得到的规划不应当看成是一成不变的，它只是在某一阶段对事物的最好认识。

BSP 方法的真正价值在于提供了两个方面的机会，一是创造一种环境和提出初步行动计划，使企业能依此对未来的系统和优先次序的改变做出反应，不致造成设计的重大失误；二是定义信息系统的职能，并不断完善。

2）BSP 方法实施步骤

BSP 的经验说明，除非得到了最高领导者和某些最高管理部门参与的承诺，否则，不要贸然开始 BSP。因为这项工作必须反映最高领导者关于企业的观点，其成果取决于管理部门能否向项目组提供企业的现状，他们对于企业的理解和对信息的需求。因此，在一开始时就要对项目的范围和目标、应交付的成果取得一致意见，避免事后的分歧，这是至关重要的。

在取得领导赞同以后，最重要的是选择项目组组长，要有一位企业领导用全部时间参加项目工作并指导项目组的活动。要确认参与研究的其他层次领导是否合适，并能正确地解释由他们所在部门得到的材料。

除了项目确立和准备工作外，BSP 还包含以下 11 个主要活动。

（1）开始。BSP 首项活动是企业情况介绍，全体项目组成员要参加。重点内容有三方面：由管理部门负责人再次重申项目的目标，期望的成果和远景规划，以及与业务活动的关系；讨论有关企业的决策过程、组织职能、关键人物、存在问题、开发策略、敏感问题，以及用户对数据处理工作的支持等；由信息系统负责人和管理人员互相介绍本

部门的情况，以便互相了解企业业务和信息化的历史和现状、目前的主要活动、计划中的变化和主要存在的问题。

（2）定义业务过程。业务过程被定义为在企业资源管理中所需要的、逻辑上相关的一组决策活动。这些活动将作为安排同管理人员面谈、确定信息总体结构、分析问题、识别数据类以及随后许多项目的基础。

（3）定义数据类。业务过程被定义后，即要识别和定义由这些过程产生、控制和使用的数据。数据指支持企业所必要的逻辑上相关的数据，即数据按逻辑相关性归成类，这样有助于数据库的长期开发。

（4）分析现存系统支持。弄清目前的数据处理如何支持企业，进而对将来的行动提出建议。对目前存在的组织、业务过程、数据处理和数据文件进行分析，发现不足和冗余，明确责任，并进一步增进对业务过程的理解。

（5）确定管理部门对系统的要求。BSP 方法必须考虑管理人员对系统的要求，并通过与高层管理人员的对话来确认项目组已做的工作，明确目标、问题、信息需求和它们的价值，并建立同最高管理部门的联系，争取他们的参与，使 BSP 项目组和管理部门间建立新的、更密切的关系。

（6）提出判断和结论。通过与管理部门的会谈对所收集的材料作出确认、解释和补充。要对问题进行分析并联系到业务过程，以便指导安排项目的优先顺序，并指明信息的改进将有助于解决问题。

（7）定义信息总体结构。定义信息总体的结构是由对目前情况的工作转向对将来计划的综合的主要步骤。信息总体结构刻画出将来的信息系统和相应的数据，使系统和它们产生的数据结构化和条理化。由于此项工作是描绘将来信息系统的蓝图，因此全体项目组成员都要加以重视。

（8）确定总体结构中的优先顺序。项目组要确定系统和数据库开发优先顺序。对信息总体结构中的子系统的项目进行排列，然后根据确定的准则评定项目的重要性，从而确定开发顺序。

（9）评价信息资源管理工作。为了使信息系统能高效率地开发、实施和运行，必须建立一个可控的环境。同时，信息系统的开发和运行过程必须加以优化，使其不断地随着技术和业务战略的变化而改变。

（10）制定建议书和开发计划。开发计划是帮助管理部门对所建议的项目作出决策，这些项目由总体结构优先顺序和信息管理部门的建议来决定。开发计划要确定具体的资源、日程和其他项目间的关系，并需估算工作规模，以便管理部门进行调度。

（11）工作成果报告。最后，项目组要向最高管理部门提交工作报告，期待最高管理部门的审查批准。

2．关键成功因素法

CSF 方法是由麻省理工学院的一个研究小组开发的，用于信息系统规划的一个有效

方法。该方法能够帮助组织找到影响系统成功的关键因素，进行分析以确定组织的信息需求，从而为管理部门控制信息技术及其处理过程提供实施指南。

在每个企业中都存在着对企业成功起关键性作用的因素，称为关键成功因素。关键因素通常与那些能够确保组织生存和发展的方面相关。不同的业务活动，关键成功因素不同。即使同一类型的业务活动在不同时期的信息需求也可能不尽相同，同一信息系统的信息需求在不同时期也不相同。

关键成功因素法能够抓住主要矛盾，使目标的识别重点突出，为管理者提供一个结构化的方法，帮助企业确定其关键成功因素和信息需求。

1）关键成功因素的确定

关键成功因素通常具有以下特征。

（1）内部 CSF：针对机构的内部活动，例如，改善产品质量、提高工效等。

（2）外部 CSF：与机构的对外活动有关，例如，满足客户企业的进入标准、获得对方的信贷等。

（3）监控型 CSF：对现有业务流程等进行监控，例如，监测零件缺陷百分比等。

（4）建设型 CSF：适应组织未来变化的有关活动，例如，改善产品组合等。

关键成功因素的来源如下：

（1）由于企业所处行业不同，而致每个行业都有一套由自己特性所决定的 CSF。

（2）由于企业的竞争策略、行业地位及地理位置的不同；即使同一行业中的企业，不同的企业也会处于不同地位；同一行业中不同企业的地理位置或竞争策略也会不同。所有这些不同，决定了一个企业可能具有不同于任何其他企业的 CSF。

（3）由于企业的外部环境因素的变化，例如，GDP 的变化、经济的波动、自然灾害、突发事件等均会引起 CSF 的改变。

（4）有时，临时因素，例如，组织内高层管理者的人事变动，也会诱致 CSF 出现暂时性的变化。

（5）由于管理者的职别、管理层级的不同，不同的管理者各有自己关心的 CSF。

CSF 共分 4 层，分别是行业的 CSF、组织的 CSF、部门的 CSF、管理者的 CSF，它们依次相互影响。可以通过内外渠道收集的数据按一定方法来验证 CSF，不容易量化的 CSF 则多由管理者做出主观判断。若要用客观方法来量度，需相当高的创意，例如，使用德尔斐法或其他方法把不同人设想的关键因素综合起来。行业关键成功因素是在竞争中取胜的关键环节，可以通过层次分析法识别行业关键成功因素。

2）CSF 实施步骤

CSF 法通过与管理者特别是高层管理者的交流，根据企业战略确定的企业目标，识别出与这些目标相关的关键成功因素及其关键性能指标。CSF 方法能够直观地引导高层管理者厘清企业战略、信息化战略与业务流程之间的关系。

CSF 实施过程通常是：通过集成高层管理者的目标而确定成功因素，通过个人的成功因素的汇总，导出组织整体的决定性成功因素，然后据此建立能够提供与这些成功因

素相关的信息系统。

第一步：了解组织的战略目标。

第二步：识别所有成功因素。可以通过与高级管理层进行交流，辨别其目标以及由此产生的成功因素；也可以采用逐层分解的方法，引出影响系统战略目标的各种因素以及影响这些因素的子因素。

第三步：确定关键成功因素。对所有成功因素进行评价，根据组织的现状和目标确定关键成功因素。

第四步：识别绩效指标和标准，以及测量绩效的数据。即给出每个成功因素的绩效指标和标准，以及用以衡量相应指标的数据。

关键成功因素与组织战略规划密切相关，组织战略描述组织期望的目标，关键成功因素则提供达到目标的关键路径和所需的测量标准。关键成功因素是为确保业务过程的成功需要完成最重要的工作，是业务过程的可观察、可测量的特征。它分布于组织的战略层、战术层、应用层及组织的各个方面，因此，需要对关键成功因素进行认真选择和度量，并对关键成功因素之间的关系进行动态调整。

3）CSF 的优缺点

CSF 的主要优点如下：

（1）管理者必须面对环境的变化，在对环境分析的基础上认真考虑如何形成自己的信息需求，对于高层管理和开发 ESS、DSS 尤其适用。

（2）该方法要求高层管理就评价标准达成共识。

CSF 的主要缺点如下：

（1）数据的汇总和数据分析过程比较随意，缺乏一种专门严格的方法将众多个人的关键成功因素汇总成一个明确的整个组织的成功因素。

（2）由于个人和组织的成功因素往往并不一致，两者之间的界限容易被混淆，从而容易使组织的成功因素具有个人倾向性。

（3）由于环境和管理经常迅速变化，信息系统也必须做出相应调整，而用 CSF 法开发的系统可能无法适应变化了的环境。

（4）CSF 在应用于较低层的管理时，由于不容易找到相应目标的关键成功因子及其关键指标，效率可能会比较低。

3．战略目标集合转化法

SST 方法将组织的战略看成是一个“信息集合”，包括使命、目标、战略和其他战略变量，如管理水平、发展趋势以及重要的环境约束等。战略性系统规划就是把组织的战略集合转化为信息系统的战略集合，而后者由信息系统的系统目标、环境约束和战略规划组成。该方法的步骤如下。

第一步：识别和阐明组织的战略集合。首先考察组织是否有书面的战略规划，如果没有，就要去构造这种战略集合。其构造过程如下：

（1）描绘出组织各类人员结构，例如，卖主、经理、雇员、供应商、顾客、贷款人、政府代理人、地区社团及竞争者等。

（2）识别每类人员的目标。

（3）对于每类人员识别其使命及战略。

第二步：将组织的战略集合转化为信息系统战略集合。这个转换过程包括对组织战略集合的每一个元素确定对应的信息系统战略元素。信息系统战略集合由系统目标、系统约束和系统设计原则组成。然后提炼出整个信息系统的结构，最后，选出一个较优的方案呈送管理层。

SSF 方法的示意图如图 12-1 所示。

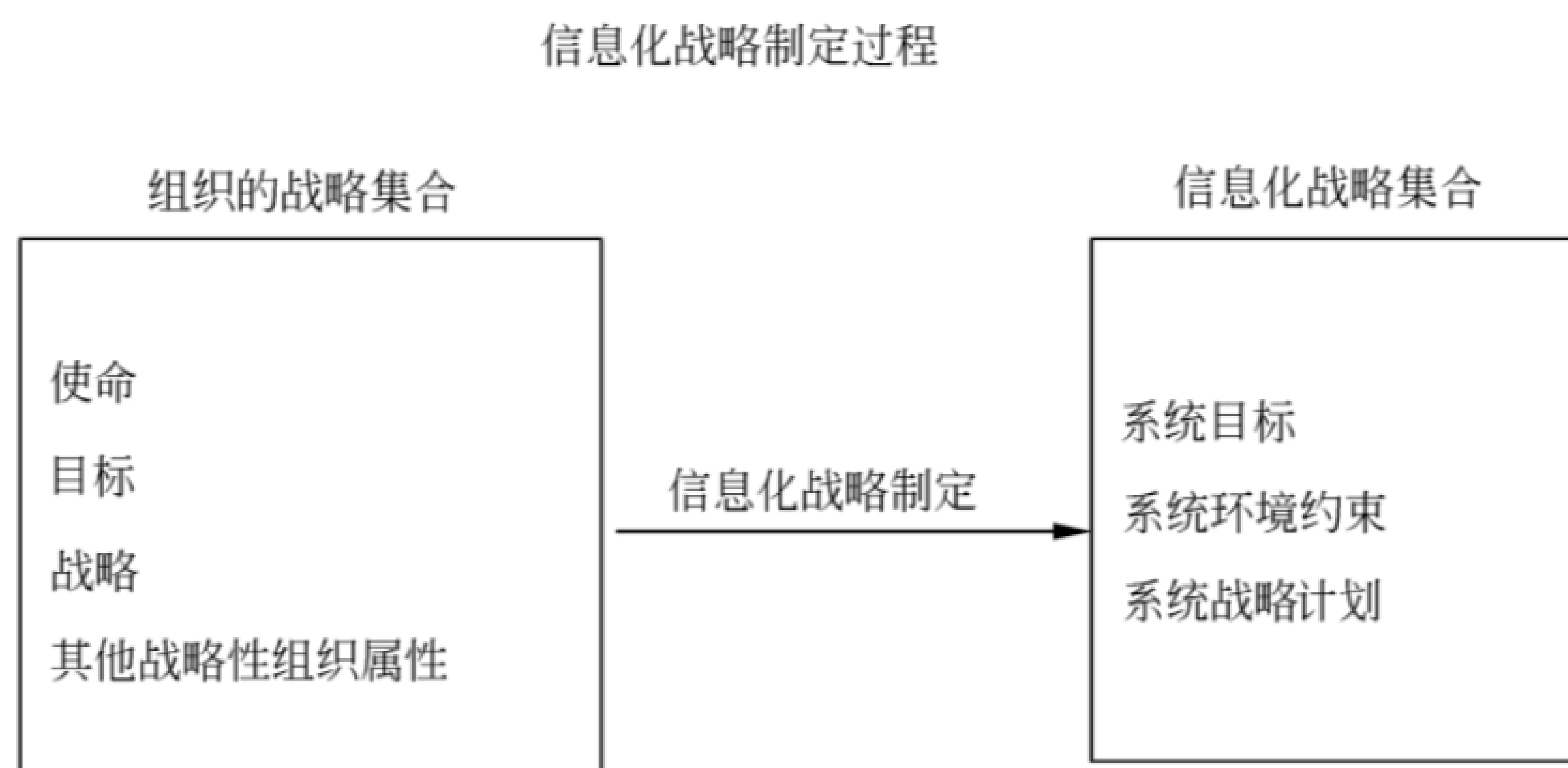


图 12-1 SSF 方法的示意图

战略目标集合转化法所描述的是从组织的基本宗旨出发，得到对系统开发阶段的输入，其目的是产生一个与组织的战略和能力紧密相关的系统。但是由于不同组织的战略目标集的内容差别很大，所以转化过程还不能形成形式化的算法。

12.2 企业信息系统建设

信息系统（Information System，IS）一般泛指收集、存储、处理和传播各种信息的具有完整功能的集合体。在这里，信息系统并没有强调收集、存储、处理和传播信息所用的工具。作为一般意义上的信息系统，在任何时代、任何社会都会存在，然而，只有到了今天，信息系统的概念才被创造出来，并得到相当程度的普及，这是因为，在当今社会，信息系统总是与计算机技术和互联网技术的应用联系在一起，因此，现代的信息系统总是指以计算机为信息处理工具，以网络为信息传输手段的信息系统。因此，现今只要说到信息系统，一般来说，就是指的这样的信息系统，而不必特意说明是“现代”信息系统。

12.2.1　信息系统的发展阶段

现代信息系统与 60 年来计算机技术和网络技术的发展保持同步。随着社会的进步和技术的发展，信息系统的内容和形式也都在不断发生着巨大的变化。与其他事物一样，信息系统也经历了一个从低级到高级、从局部到全局、从简单到复杂的发展过程。信息系统大致经历了 4 个发展阶段。

1．第一阶段：电子数据处理阶段

计算机应用于企业是从简单数据处理开始的。计算机发明以后的一段时期，计算机仅仅用于科学计算。后来，计算机程序设计人员将计算机应用领域进行了拓展，开始尝试用计算机进行数据处理，从而开辟了计算机的更广阔的应用领域。不过，最早的计算机在数据处理中的应用，仅着眼于减轻人们在计算方面的劳动强度。例如，用于计算工资、统计账目等，属于电子数据处理业务，对企业单项业务进行处理，较少涉及管理内容。

2．第二阶段：事务处理阶段

随着企业业务需求的增长和技术条件的发展，人们逐步将计算机应用于企业局部业务的管理，如财会管理、销售管理、物资管理、生产管理等，即计算机应用发展到对企业的局部事务的管理，形成了所谓事务处理系统（Transaction Process System，TPS），但它并未形成对企业全局的、整体的管理。

3．第三阶段：管理信息系统阶段

人们常说的信息系统大多指支持各部门和机构管理决策的信息系统，因此，信息系统一般又称为管理信息系统（Management Information System，MIS）。人们从不同的角度对它进行了定义，比较被广泛认可的定义是："管理信息系统是用系统思想建立起来的，以电子计算机为基本信息处理手段，以现代通信设备为基本传输工具，且能为管理决策提供信息服务的人机系统。即管理信息系统是一个由人和计算机等组成的，能进行管理信息的收集、传输、存储、加工、维护和使用的系统"。

在 MIS 阶段，信息系统形成了对企业全局性的、整体性的计算机应用。MIS 强调以企业管理系统为背景，以基层业务系统为基础，强调企业各业务系统间的信息联系，以完成企业总体任务为目标，它能提供企业各级领导从事管理需要的信息，但其收集信息的范围还更多地侧重于企业内部。

4．第四阶段：决策支持系统阶段

当前，计算机信息系统已经从 MIS 发展成更强调支持企业高层决策的决策支持系统（Decision Surport System，DSS），即决策支持系统阶段。

Internet 技术的发展和应用，在很大程度上拓展和提升了信息系统的功能和作用，其最大的特点是通过 Internet 众多的孤立的信息系统（即信息孤岛）加以联系起来，形成

在更大程度上实现信息共享的大范围的基于网络互联的信息系统。Internet 技术应用于企业内部信息系统，可以促进企业内综合 MIS、DSS 功能，并以办公自动化技术为支撑的办公信息系统的实施。企业信息系统的目标为：借助于自动化和互联网技术，综合企业的经营、管理、决策和服务于一体，以求达到企业和系统的效率、效能和效益的统一，使计算机和互联网技术在企业管理和服务中能发挥更显著的作用。

希赛教育专家提示：信息系统的 4 个发展阶段，它们之间的关系并不是取代关系，而是互相促进、共同发展的关系，也就是说，在一个企业里，以上 4 个阶段的信息系统，可能同时都存在，也可能只有其中的一种、两种或三种。更高级的是几种信息系统互相融合成一体。

12.2.2 信息系统的功能

信息系统的功能就是信息系统的使用价值，也就是信息系统所能做的事情和所起的作用。信息系统的功能是一个“多面体”，即从不同的角度分析，其功能是不一样的。

1．需求功能和实现功能

从企业业务需求的角度看，信息系统的功能可以分为需求功能和实现功能。所谓需求功能，是企业的业务对信息系统提出的要求，例如，一个生产零售产品的企业，它非常需要如该产品的市场容量，竞争对手的情况等信息；而实现功能则是指，对于业务需求信息，由于各种客观条件的限制，信息系统只能提供其中的一部分，这部分就是实现功能。

2．初级功能和高级功能

从发展阶段的角度看，可分为初级功能和高级功能，初级功能逐步发展为高级功能。在这方面，有很多人做了非常深入的研究，具有代表性的是著名的诺兰（Nolan）模型。Nolan 在 1973 年提出了信息系统发展的阶段理论，被称为诺兰阶段模型。到 1980 年，诺兰进一步完善该模型，把信息技术的成长过程划分为 6 个阶段。

第一阶段为初装（Initiation）阶段。它的标志是组织安装了第一台计算机并引入了自动化概念，同时初步开发了管理应用程序。该阶段，计算机的作用被初步认识。一般大多发生在财务部门。

第二阶段为蔓延（Contagion）阶段。其标志是随着自动化的扩展（从少数部门扩散到多数部门，并开发了大量的应用程序）而导致的计算机系统的急增。在该阶段，数据处理能力发展得最为迅速，但同时也出现了许多亟待解决的问题，例如，数据冗余性、不一致性、难以共享等。

第三阶段为控制（Control）阶段。其标志是试图遏制快速上升的计算机服务成本并将数据处理置于控制之下。为了加强组织协调，出现了由企业领导和职能部门负责人参加的领导小组，对整个企业的系统建设进行统筹规划，特别是利用数据库技术解决数据共享问题。

第四阶段为集成（Integration）阶段。其标志是各种各样的系统和技术集成为内在统一的系统，数据处理发展进入再生和控制发展时期。

第五阶段为数据管理（Data administration）阶段。其标志是完全集成的、基于数据的系统发展和实施的结束。

第六阶段为成熟（Maturity）阶段。其标志是企业数据管理的日益成熟，可以满足单位中各管理层次的要求，从而实现信息资源的管理。

随后，Nolan 又将该模型的 6 个阶段划分为两个时期，即计算机时代和信息时代，其中，前三个阶段构成计算机时代，后三个阶段进入信息时代。Nolan 模型能够帮助一个组织识别其所处的阶段从而确立相应的发展战略，是信息化战略管理的重要理论工具。

Nolan 的两个时代划分理论于 20 世纪 90 年代之后逐渐不能适应信息技术发展的需要，为此，Nolan 又提出了一种理解组织内部信息技术演化的新框架，该框架将信息技术的发展分为三个阶段，即数据处理阶段、信息技术阶段和网络阶段。

在数据处理阶段（20 世纪 60～80 年代），信息技术主要在一个组织的操作层面和管理层面起作用，其主要功能是使一些专门的工作自动化；在信息技术阶段（80 年代～90 年代中期），信息技术在一个组织中的发展进入战略管理层面，强调知识工作者对信息技术的利用；在网络阶段（20 世纪 90 年代中期之后），信息技术不再能够单方面地使组织取得他们所寻求的业务效果，信息技术与组织人员及其工作整合为一种网络化的组织形式以创造 10 倍速的生产率。

Nolan 模型提出以后不久，美国学者 Edgar Schein 也提出了一种称之为“新的信息技术发展阶段模型”的理论，其特点是将信息技术演化过程与组织变迁过程联系起来考察，有助于形成一种整体化的认识。该模型包括 4 个阶段。

第一阶段是投资或启动阶段。组织决定在新的信息技术方面投资，新技术能够带来明显的益处则将顺利进入第二阶段，如果该阶段没有用户参与或发生了供应商方面的问题，那么就会延迟信息技术的演化，并导致成本超出预算、项目缺乏管理及其他不可预期的技术问题。

第二阶段是技术学习和适应阶段。用户通过学习如何利用技术来完成任务，如果用户有机会更好地理解新技术及其益处则顺利进入第三阶段，如果过早地控制技术发展，那么就会影响用户的学习过程，并导致缺乏进一步开发信息技术潜力的动机等问题。

第三阶段是管理控制阶段。组织认识到信息技术的重要性并对系统发展和实施过程予以精确地控制，如果控制过程能够确保各种应用的成本-效益的成功则顺利进入第四阶段，如果出现过多的控制，就会导致创新热情的丧失、新技术扩散的失败乃至从头再来等问题。

第四阶段是大范围的技术转移阶段。新的信息技术如局域网技术等将转移到组织的其他部门，信息技术知识也将随技术向用户转移，信息技术成为组织结构的有机组成部分。

Schein 的信息技术阶段论提供了 IT 角色的一种新视角，其主要作用仍在于引导战略信息管理者识别组织所处的信息技术发展阶段并结合所采用的信息技术类型制定针对性的发展战略和方案，它同时也表明，不适当的、过早的和过多的控制不利于信息技术的应用和普及，并容易导致各种停滞问题。

3．通用功能和专业功能

任何一个信息系统都必须具备一些基本的、通用功能。同时，作为一个信息系统，都必然是在特定环境下产生，而且为了实现特定的目标，因此，每个信息系统一般还要具有一些特有的、专业的功能。

一个信息系统应当具有的通用功能一般来说，是多方面的，而且是复杂的，但是，以下基本功能是任何一个信息系统所必不可少的：数据库功能、存储信息功能、检索信息的功能、信息分析功能。

4．整体功能和局部功能

建设信息系统是企业信息化非常重要的和关键的内容。因此，信息系统必须能够体现企业的总体战略，也就是说，企业信息系统必须为企业总体战略服务，这就是信息系统的整体功能。同时，一个信息系统必然由若干个部分组成，即可以分为多个子系统，而每一个子系统都要有自己的功能，这种功能就是局部功能。

为了使信息系统的整体功能和每个部分的局部功能都能得到加强，有必要在整体规划的指导下，建立起全企业信息系统功能模型，然后，再根据各子系统和程序模块的具体情况，对各个局部功能进行优化、整合，从而形成在市场竞争中具有快速反映能力的、完善的信息系统。

12.2.3 信息系统的类型

当今的信息系统，由于其广泛的应用，已经发展成为一个极为庞大的家族，而且几乎每个信息系统其内部构成都非常复杂。为了充分认识信息系统，有必要对其进行分类。但是，如何进行分类，并不是一个简单的问题。目前对于信息系统有很多的分类方法，例如，从计算机应用的角度，可以分成人工信息系统和基于计算机的信息系统；从独立性的角度，可以分成独立信息系统和综合信息系统；从处理方式的角度，可以分成批处理信息系统和联机处理信息系统等。下面介绍两种重要的信息系统分类方法。

1．以数据环境分类

目前，对于信息系统最为权威的分类方法是詹姆斯·马丁的分类。马丁从信息系统的数据环境的角度出发，对信息系统进行分类。

马丁在《信息工程》和《战略数据规划方法学》中将信息系统的数据环境分为 4 种类型，并认为清楚地了解它们之间的区别是很重要的，因为它们对不同的管理层次（包括高层管理）的作用是不同的。

第一类数据环境是数据文件。这类信息系统没有使用数据库管理系统，根据大多数的应用需要，由系统分析师和程序员分散地设计各种数据文件。其特点是简单，相对容易实现。但随着应用程序增加，数据文件数目剧增，导致很高的维护成本；一小点应用上的变化都将引起链锁反应，使修改和维护工作既缓慢成本又高昂，并很难进行。

第二类数据环境是应用数据库。这类信息系统虽然使用了数据库管理系统，但没达到第三类数据环境那种共享程度。分散的数据库为分散的应用而设计。实现起来比第三类数据环境简单。像第一类数据环境一样，随着应用的扩充，应用数据库的个数，以及每个数据库中的数据量也在剧烈地增加，随之而导致维护成本大幅度增高，有时甚至高于第一类数据环境。该类数据环境还没有发挥使用数据库的主要优越性。

第三类数据环境是主题数据库（Subject DataBase）。这类信息系统所建立的一些数据库与一些具体的应用有很大的独立性，数据经过设计，其存储的结构与使用它的处理过程都是独立的。各种面向业务主题的数据，例如，顾客数据、产品数据或人事数据，通过一些共享数据库被联系和体现出来。这种主题数据库的特点是：经过严格的数据分析，建立应用模型，虽然设计开发需要花费较长的时间，但其后的维护成本很低。最终（但不是立即）会使应用开发加快，并能使用户直接与这些数据库交互使用数据。主题数据库的开发需要改变传统的系统分析方法和数据处理的管理方法。但是，如果管理不善，也会蜕变成第二类甚或是第一类数据环境。

第四类数据环境是信息检索系统。一些数据库被组织得能保证信息检索和快速查询的需要，而不是大量的事务管理。软件设计中要采用转换文件、倒排表或辅关键字查询技术。新的字段可随时动态地加入到数据结构中。有良好的最终用户查询和报告生成软件工具。大多数用户掌握的系统都采用第四类数据库。这种环境的特点是：比传统的数据库有更大的灵活性和动态可变性。一般应该与第三类数据环境共存，支持综合信息服务和决策系统。

2．以应用层次分类

企业的管理活动可以分成 4 级，分别是战略级、战术级、操作级和事务级，相应地，信息系统就其功能和作用来看，也可以分为 4 种类型，即战略级信息系统、战术级信息系统、操作级信息系统和事务级信息系统。不同级别的信息系统的所有者和使用者都是不同的。一般来说，战略级的信息系统的所有者和使用者都是企业的最高管理层，对于现代企业制企业，就是企业的董事会和经理团队；战术级信息系统的使用者一般是企业的中层经理及其管理的部门；操作级信息系统的使用者一般是服务型企业的业务部门，例如，保险企业的保单处理部门；事务级信息系统的使用者一般是企业的管理业务人员，例如，企业的会计、劳资员等。

12.2.4　信息系统建设的复杂性

大型信息系统的建设是资金密集、技术密集的宏大而复杂的系统工程，它的复杂性

不仅来自于计算机、网络和通信等一系列现代技术方面的因素，更重要的是来自于系统建设和管理体制方面的关系和联系，还来自于企业之外的社会因素。因此，信息系统的建设侧重于技术的系统工程，例如，大型电站、大型工厂等有很大区别，通常一项技术工程通过规划设计、研制生产、安装调试后即可进入稳定的运行。而信息系统则不然，其复杂性既由于技术的复杂，又由于管理的复杂，而且当两者结合起来以后，其复杂性就尤其突出。其系统需求和内外部条件总是处于不断的变化之中。因此，没有一个信息系统建好后是一劳永逸的，系统的修改、维护、升级、扩展，甚至是重建都是会经常发生的。

1．信息系统开发的复杂性

信息系统是一个公认的复杂系统，其复杂性既由于技术的复杂，又由于管理的复杂，而且当两者结合起来以后，其复杂性就尤其突出。

技术复杂性是多方面的，一是信息系统涉及到的技术跨越多个领域，有计算机科学与技术领域，包括软件、硬件等；通信领域，包括有线通信、无线通信等；网络领域，包括局域网、广域网、内联网、外联网、互联网等；以及数学等。因此，在信息系统的开发中，如何选择技术，以及选择什么样的技术，无疑都有很多难题。二是信息系统所用到的技术都是当今的热门技术，其发展变化异常迅速，很多技术的生命周期很短，有些技术的生命周期一二年，也有的技术刚刚出生不久，就面临被更新的技术所替代的命运。面对层出不穷的新技术，信息系统的开发者必须作远见卓识的判断。三是与新技术相适应的新产品也是层出不穷，令人目不暇接。对此，信息系统的开发者要作出准确的选择。

管理方面的复杂性更为突出。在开始开发的初期，一般来说，很难给出企业信息系统的一个明确轮廓，究竟要建成一个什么样的信息系统，对管理人员来说还都是一个谜，即使开发者设计出来一个完整的方案，也很难向领导及业务人员解释清楚，说服所有的人。至于信息系统将会带来什么效益，更难明确回答。

2．信息系统运行的复杂性

信息系统开发的复杂性是很大的，但是，运行的困难性可能会更大，这是因为，信息系统的运行需要有科学的管理体制、良好的管理基础、完善的管理机构、合理的管理流程，还要有管理人员，尤其是领导的支持和参与。而要做到这些，就要首先解决管理上的问题，例如管理体制和管理机构的调整、业务流程的优化或重组，管理人员习惯观念等的改变，等等。

一个牵涉到企业全局的信息系统要做到良好地运行，需要特别解决好以下 4 个问题。

一是要解决基础数据的问题。一个信息系统所处理的对象主要是数据，因此，数据的质量问题是十分重要的。软件工程中有一句话：“输入的是垃圾，输出的肯定也是垃圾”。这就是说，信息系统不可能“化腐朽为神奇”，不可能把垃圾数据处理成有用的数据。而一些信息系统的需求单位，恰恰是基础数据不全、不准确或不一致。所谓数据

不全，是指只有部分信息系统所需要的数据，例如，一个企业有 10 个下属单位，只有 6 个下属单位有数据，其他则没有，这样一来，该系统的运行效果就必然大打折扣；所谓数据不准确，是指一些基础数据有差错，由此，必然影响系统的可靠性；所谓数据不一致，是指同一项数据在不同的地方取不同的值。

二是领导介入的问题。企业的信息系统绝不仅仅是一个软件的使用，它要涉及到企业的组织流程，涉及到企业的机构调整，涉及到因信息系统的运行而使企业发生许多新的变化，这些都决定了信息系统不是一个技术的问题；同时，许多问题和障碍也不是仅靠技术人员就能解决的。信息系统的运行需要企业最高领导层的介入，而在一些企业的管理层里，对此却缺乏足够的认识，在一些企业里，最高管理层把信息系统的建设和运行交给信息技术部门就算万事大吉，持有这样做法的企业，其信息系统的良好运行将成问题。

三是最终用户问题。企业信息系统的最终用户，也就是信息系统的使用者往往是那些企业管理机构的业务人员。信息系统运行的难题是要让这些业务人员接受信息系统，首先，需要改变他们长时间形成的一些工作习惯，这往往是比较困难的，再者，这些业务人员需要熟悉并掌握信息系统的一些技术和工作方法，这也需要一个比较复杂的过程。

四是系统分析师。信息系统是复杂的人机工程，因此，最需要的人才是既懂经营管理又懂计算机技术的专家型的人才，也就是系统分析师。而很多企业在建设和运行信息系统时，恰恰缺少的就是系统分析师。

3．信息系统维护改造的复杂性

由于企业内外部环境和企业经营管理需求的不断变化，信息系统的维护改造经常是不可避免的，特别是随着计算机设备的不断降价，个人计算机越来越多地出现在管理人员的办公桌上，要发挥这些设备的效益，必须把它们互联起来，既要满足每个管理人员的信息需要，又要给高层领导提供及时的决策信息。这时，人们才吃惊地发现，分散的开发所带来的严重后果：修改原先的软件，重新组织数据，连成一个统一的大系统，所耗费的人力和资金比重新建立还要多；甚至，采取维护和修改的办法是根本行不通的。系统维护问题就像病魔似的缠住了数据处理的发展，这就是人们所说的“数据处理危机”。传统的数据处理开发方法所遭到的一些失败，也是这种危机的表现。

12.2.5　信息系统的生命周期

信息系统与其他事物一样，也要经历产生、发展、成熟和消亡的过。把信息系统从产生到消亡的整个过程称为信息系统的生命周期。一般来说，信息系统的生命周期分为 4 个阶段，即产生阶段、开发阶段、运行阶段和消亡阶段。

1．信息系统的产生阶段

信息系统的产生阶段，也是信息系统的概念阶段或者是信息系统的需求分析阶段。这一阶段又分为两个过程，一是概念的产生过程，即根据企业经营管理的需要，提出建

设信息系统的初步想法；二是需求分析过程，即对企业信息系统的需求进行深入地调研和分析，并形成需求分析报告。

2．信息系统的开发阶段

信息系统的开发阶段是信息系统生命周期中最重要和最关键的阶段。该阶段又可分为 5 个阶段，分别是总体规划、系统分析、系统设计、系统实施和开发评价阶段。

（1）总体规划阶段。信息系统总体规划是系统开发的起始阶段，它的基础是需求分析。以计算机和互联网为工具的信息系统是企业管理系统的重要组成部分，是实现企业总体目标的重要工具。因此，它必须服从和服务于企业的总体目标和企业的管理决策活动。总体规划的作用主要有以下一些：

- 指明信息系统在企业经营战略中的作用和地位。
- 指导信息系统的开发。
- 优化配置和利用各种资源，包括内部资源和外部资源。
- 通过规划过程规范企业的业务流程。

一个比较完整的总体规划，应当包括信息系统的开发目标、信息系统的总体架构、信息系统的组织结构和管理流程、信息系统的实施计划、信息系统的技术规范等。

（2）系统分析阶段。系统分析阶段的目标是为系统设计阶段提供系统的逻辑模型。

系统分析阶段以企业的业务流程分析为基础，规划即将建设的信息系统的基本架构，它是企业的管理流程和信息流程的交汇点。系统分析的内容主要应包括组织结构及功能分析、业务流程分析、数据和数据流程分析、系统初步方案等。

（3）系统设计阶段。系统设计阶段是根据系统分析的结果，设计出信息系统的实施方案。系统设计的主要内容包括系统架构设计、数据库设计、处理流程设计、功能模块设计、安全控制方案设计、系统组织和队伍设计、系统管理流程设计等。

（4）系统实施阶段。系统实施阶段是将设计阶段的结果在计算机和网络上具体实现，也就是将设计文本变成能在计算机上运行的软件系统。由于系统实施阶段是对以前的全部工作的检验，因此，系统实施阶段用户的参与特别重要。如果说在系统设计阶段以前，用户处于辅助地位的话，而到了系统实施阶段以后，用户就应逐步变为系统的主导地位。

（5）系统验收阶段。信息系统实施阶段结束以后，系统就要进入试运行。通过试运行，系统性能的优劣、是否做到了用户友好等等问题都会暴露在用户面前，这时就进入了系统验收阶段。

3．信息系统运行阶段

当信息系统通过验收，正式移交给用户以后，系统就进入了运行阶段。一般来说，一个性能良好的系统，运行过程中会较少出现故障，即使出现故障，也较容易排除；而那些性能较差的系统，运行过程中会故障不断，而且可能会出现致命性故障，有时故障会导致系统瘫痪。因此，长时间的运行是检验系统质量的试金石。

另外，要保障信息系统正常运行，一项不可缺少的工作就是系统维护。在软件工程

中，把维护分为 4 种类型，即纠正性维护、适应性维护、完善性维护和预防性维护。一般在系统运行初期，纠正性维护和适应性维护比较多，而到后来，完善性维护和预防性维护就会比较多。有关系统维护方面的详细知识，请阅读第 6.1 节。

4．信息系统消亡阶段

通常，人们比较重视信息系统的开发阶段，轻视信息系统运行阶段，而几乎完全忽视信息系统的消亡阶段。其实，这样做是片面的。因为，计算机技术和互联网技术的发展十分快速，新的技术、新的产品不断出现；同时，由于企业处在瞬息万变的市场竞争的环境之中，在这种情况下，企业开发好一个信息系统想着让它一劳永逸地运行下去，是不现实的。企业的信息系统会经常不可避免地会遇到系统更新改造、功能扩展，甚至是报废重建的情况。对此企业应当在信息系统建设的初期就要注意系统的消亡条件和时机，以及由此而花费的成本。

12.2.6　信息系统建设的原则

为了能够适应开发的需要，在信息系统规划设计，以及系统开发的过程中，必须要遵守一系列原则，这是系统成功的必要条件。下面几条原则是信息系统开发常用的原则。

1．高层管理人员介入原则

一个信息系统其建设的目标总是为企业的总体目标服务的，否则，这个系统就不应当建设。而真正能够理解企业总体目标的人必然是那些企业高层管理人员，只有他们才能知道企业究竟需要什么样的信息系统，而不需要什么样的信息系统，也只有他们才知道企业有多大的投入是值得的，而超过了这个界限就是浪费。这点是那些身处某一部门的管理人员，或者是技术人员所无法做到的。因此，信息系统从概念到运行都必须有企业高层管理人员介入。当然，这里的“介入”有着其特定的含义，它可以是直接参加，也可以是决策或指导，还可以是在政治、经济、人事等方面的支持。

希赛教育专家提示：高层管理人员介入原则在现阶段已经逐步具体化，那就是 CIO 的出现。CIO 负责企业信息化的工作，主持制定企业信息规划、政策、标准，并对全企业的信息资源进行管理控制的企业行政官员。在大多数企业里，CIO 是企业最高管理层中的核心成员之一。毫无疑问，深度介入信息系统开发建设，以及运行是 CIO 的职责所在。有关 CIO 的详细知识，请阅第 12.7 节。

2．用户参与开发原则

在我国，流行着信息系统开发中所谓“用户第一”或是“用户至上”的原则。当然，这个原则并没有错，一个成功的信息系统，必须把用户放在第一位，这应该是毫无疑问的。但是，究竟应当怎么“放”？怎么“放”才算是第一位？都没有一个确切的标准。而马丁提出的“用户参与开发原则”就把“用户第一原则”具体化了。

用户参与开发原则主要包括以下几项含义：

一是“用户”是有确定的范围。究竟谁是用户？通常把“用户”仅仅理解成为用户

单位的领导，其实，这是很片面的。当然，用户单位领导应该包括在用户范围之内，但是，更重要的用户或是核心用户是那些信息系统的使用者，而用户单位的领导只不过是辅助用户或是外围用户。

二是用户特别是那些核心用户，不应是参与某一阶段的开发，而应当是参与全过程的开发，即用户应当参与从信息系统概念规划和设计阶段，直到系统运行的整个过程。而当信息系统交接以后，他们就成为系统的使用者。

三是用户应当深度参与系统开发。用户以什么身份参与开发是一个很重要的问题。一般说来，参与开发的用户人员，既要以甲方代表身份出现，又应成为真正的系统开发人员，与其他开发人员融为一体。

3．自顶向下规划原则

在信息系统开发的过程中，经常会出现信息不一致的问题，这种现象的存在对于信息系统来说往往是致命的，有时，一个信息系统会因此而遭到报废的结果。研究表明，信息的不一致是由计算机应用的历史性演变所造成的，它通常发生在没有一个总体规划的指导，就来设计实现一个信息系统的情况之下。因此，坚持自顶向下规划原则对于信息系统的开发和建设来说是至关重要的。自顶向下规划的一个主要目标是达到信息的一致性。同时，自顶向下规划原则还有另外一个方面，那就是这种规划绝不能取代信息系统的详细设计。必须鼓励信息系统各子系统的设计师在总体规划的指导下，进行有创造性的设计。

4．工程化原则

在 20 世纪 70 年代，出现了世界范围内的“软件危机”。所谓软件危机，是指一个软件开发好以后，谁也无法保证它能够正确地运行，也就是软件的可靠性成了问题。软件危机曾一度引起人们特别是工业界的恐慌。经过探索，人们认识到，之所以会出现软件危机，最主要的原因是，软件产品是一种个体劳动产品，最多也就是作坊式的产品。因此，没有工程化是软件危机发生的根本原因。此后，发展成了“软件工程”这门工程学科，在一定程度上解决了软件危机。

信息系统也经历了与软件开发大致相同的经历。在信息系统发展的初期，人们也像软件开发初期一样，只要作出来就行，根本不管实现的过程。这时的信息系统，大都成了少数开发者的“专利”，系统可维护性、可扩展性都非常差。后来，信息工程、系统工程等工程化方法被引入到信息系统开发过程之中，才使得问题得到了一定程度的解决。

其实，工程化不仅是一种有效的方法，它也应当是信息系统开发的一项重要原则。

5．其他原则

对于信息系统开发，人们还从不同的角度提出了一系列原则，例如：

（1）创新性原则，用来体现信息系统的先进性。

（2）整体性原则，用来体现信息系统的完整性。

（3）发展性原则，用来体现信息系统的超前性。

（4）经济性原则，用来体现信息系统的实用性。

12.2.7　信息系统开发方法

企业信息系统对于企业信息化的重要意义是不言而喻的。从实际运行的效果来看，有些信息系统运行得很成功，取得了巨大的经济效益和社会效益；但也有些信息系统效果并不显著，甚至还有个别信息系统开始时还能正常运行，可时间一长，系统就故障不断，最后走上报废之路。是什么导致这样截然不同的结果呢？当然，这里的原因可能很复杂，但有一个原因是十分重要和关键的，那就是信息系统的开发方法问题。

信息系统是一个极为复杂的人-机系统，它不仅包含计算机技术、通信技术，以及其他的工程技术，而且，它还是一个复杂的管理系统，还需要管理理论和方法的支持。下面简单介绍几种最常用的信息系统开发方法。

1．结构化方法

结构化方法是由结构化系统分析和设计组成的一种信息系统开发方法，本书第 4.1 节详细地介绍了该方法。

结构化方法是目前最成熟、应用最广泛的信息系统开发方法之一。它是假定被开发的系统是一个结构化的系统，因此，其基本思想是将系统的生命周期划分为系统调查、系统分析、系统设计、系统实施、系统维护等阶段。这种方法遵循系统工程原理，按照事先设计好的程序和步骤，使用一定的开发工具，完成规定的文档，在结构化和模块化的基础上进行信息系统的开发工作。结构化方法的开发过程一般是先把系统功能视为一个大的模块，再根据系统分析设计的要求对其进行进一步的模块分解或组合。

结构化生命周期法主要特点如下：

（1）开发目标清晰化。结构化方法的系统开发遵循“用户第一”的原则，开发中要保持与用户的沟通，取得与用户的共识，这使得信息系统的开发建立在可靠的基础之上。

（2）工作阶段程式化。结构化方法每个阶段的工作内容明确，注重开发过程的控制。每一阶段工作完成后，要根据阶段工作目标和要求进行审查，这使阶段工作有条不紊，也避免为以后的工作留下隐患。

（3）开发文档规范化。结构化方法每一阶段工作完成后，要按照要求完成相应的文档，以保证各个工作阶段的衔接与系统维护工作的便利。

（4）设计方法结构化。结构化方法采用自上而下的结构化、模块化分析与设计方法，使各个子系统间相对独立，便于系统的分析、设计、实现与维护。结构化方法被广泛地应用于不同行业信息系统的开发中，特别适合于那些业务工作比较成熟、定型的系统，例如，银行、电信、商品零售等行业。

2．快速原型法

快速原型法是一种根据用户需求，利用系统开发工具，快速地建立一个系统模型展示给用户，在此基础上与用户交流，最终实现用户需求的信息系统快速开发的方法。在现实生活中，一个大型工程项目建设之前制作的沙盘，以及大型建筑的模型等都与快速

原型法有同样的功效。应用快速原型法开发过程包括系统需求分析、系统初步设计、系统调试、系统检测等阶段。用户仅需在系统分析与系统初步设计阶段完成对应用系统的简单描述，开发者在获取一组基本需求定义后，利用开发工具生成应用系统原型，快速建立一个目标应用系统的最初版本，并把它提交给用户试用、评价，根据用户提出的意见和建议进行修改和补充，从而形成新的版本，再返回给用户。通过这样多次反复，使得系统不断地细化和扩充，直到生成一个用户满意的方案为止。

快速原型法具有开发周期短、见效快、与业务人员交流方便的优点，特别适用于那些用户需求模糊，结构性比较差的信息系统的开发。

3．企业系统规划方法

BSP 方法是企业战略数据规划方法和信息工程方法的基础和，也就是说，后两种方法是在 BSP 方法的基础上发展起来的，因此，了解并掌握 BSP 方法对于全面掌握信息系统开发方法是有帮助的。BSP 方法的目标是提供一个信息系统规划，用以支持企业短期的和长期的信息需求。有关 BSP 方法的详细知识，请阅读第 12.1.4 节。

4．战略数据规划方法

詹姆斯•马丁提出的战略数据规划方法是信息系统开发极为重要的一种方法。按照马丁的观点，企业要建设信息系统，它没有必要急着去购置设备，也没有必要马上组织软件开发和上网，它的首要任务应该是在企业战略目标的指导下做好企业战略数据规划。一个好的企业战略数据规划应该是企业核心竞争力的重要构成因素，它有非常明显的异质性和专有性，必将成为企业在市场竞争中的制胜法宝。

战略数据规划方法的要点主要有以下一些：

（1）数据环境对于信息系统至关重要。企业数据环境是随着企业的发展不断变化的，也是企业发展的基础条件。信息系统建设极大影响着企业的未来发展方向，对企业的数据环境提出了更高的要求。把静态的、独立的信息资源通过战略数据规划重建企业数据环境，使其成为集成化、网络化的信息资源，对一个现代化企业来说是更为迫切的任务。

（2）4 种数据环境。在信息系统发展的历程中共有 4 类数据环境，即数据文件、应用数据库、主题数据库和信息检索系统。

（3）建设主题数据库是信息系统开发的中心任务。这里的主题数据库并不是指数据库的大小，也不是指数据库的功能是什么，而是指那些数据库是面向企业的业务主题的，那些不是面向业务主题。所谓业务主题，就是指企业的核心业务和主导流程。比如，对于一个机加工企业来说，生产机件产品就是其核心业务，相应地，围绕核心业务建立的数据库就是企业的主题数据库。而对于一个保险企业来说，围绕着保单处理的数据库就是企业的主题数据库。

（4）围绕主题数据库搞好应用软件开发。

5．信息工程方法

信息工程方法是詹姆斯•马丁创立的面向企业信息系统建设的方法和实践。信息工程方法与企业系统规划方法和战略数据规划方法是一种交叉关系，即信息工程方法是其

他两种方法的总结和提升，而其他两种方法则是信息工程方法的基础和核心。

信息工程是计算机信息系统发展到比较成熟阶段的产物，它不仅为大型信息系统的开发给出了方法和技术，而更重要的是它在理论与实践的结合上对大型信息系统的开发提出了相应的开发策略和原则，而这些策略和原则对于信息系统的成功开发和应用都是至关重要的。虽然，信息工程是在 20 世纪 80 年代末期发展起来的，但是，在今天，仍然对信息系统的开发具有重要的指导价值。

信息工程方法与信息系统开发的其他方法相比，有一点很大的不同，就是信息工程不仅是一种方法，它还是一门工程学科。它第一次把信息系统开发过程工程化了。所谓工程化，就是指有一整套成熟的、规范的工程方法、技术、标准、程序和规范，使得开发工作摆脱随意性和多变性，其目标是信息系统的开发走上智能化、程序化和自动化的道路。

6．面向对象方法

信息工程在实际应用中既表现出其优越性的一面，同时，也暴露了一些缺点，例如，过于偏重数据，致使应用开发受到影响。而面向对象方法则集成了以前各种方法的优点，避免了各自的一些缺点。

面向对象的分析方法是利用面向对象的信息建模概念，例如实体、关系、属性等，同时运用封装、继承、多态等机制来构造模拟现实系统的方法。传统的结构化设计方法的基本点是面向过程，系统被分解成若干个过程。而面向对象的方法是采用构造模型的观点，在系统的开发过程中，各个步骤的共同的目标是建造一个问题域的模型。在面向对象的设计中，初始元素是对象，然后将具有共同特征的对象归纳成类，组织类之间的等级关系，构造类库。在应用时，在类库中选择相应的类。

12.3 信息资源管理

信息孤岛是信息化过程中出现的一种现象，它是信息化的成本因素，也就是说，信息孤岛是信息化的阻碍力量，是信息化提升效率的重要瓶颈。因此，消除至少要削弱信息孤岛的影响，是信息化进程中的重要任务。

12.3.1 信息孤岛形成的原因

信息孤岛虽然屡屡被人提起，然而，并没有一个被人普遍接受的定义。信息孤岛通常是指，在一个单位的部门之间、系统之间、业务之间由于种种原因造成信息隔离，无法或很少能互相交换和交流，亦即信息不能跨部门、跨系统和跨业务流动。例如，财务信息只在财务部门生成、处理和使用，其他部门不能共享。这种现象，人们习惯性地称为信息孤岛。

信息孤岛在技术上带来的不良影响列举如下：

（1）数据的一致性无法保证。由于信息定义与采集过程彼此独立，企业的同一数据可能在不同的应用中不一致。

（2）信息无法共享和反馈。使得一些业务由于不能获得所需信息而受影响。

（3）信息需要重复多次的采集和输入。不仅仅增加额外的劳动，同时，也是数据失真的重要原因。

信息孤岛形成的原因是多方面的，包括信息化初级阶段原因、技术原因、管理原因、人员原因、标准原因等。

（1）信息化初级阶段原因。企业信息化处于初级阶段是信息孤岛形成的客观原因。因为，只有在企业实施信息化，开发应用信息系统才能形成“信息岛”，进一步讲，只有多个“信息岛”才有可能出现所谓的“孤岛”，亦即信息孤岛。

（2）技术原因。信息系统是信息技术的集成应用，因此，技术上的缺陷或初级必然会形成信息孤岛。而技术原因又可分为两种情况，一是技术不够成熟。例如，在网络未普及之前，大部分企业都是单机应用，自然形成许多信息孤岛。那时，想要消除信息孤岛必然有较大的成本支出；二是选用技术有缺陷。例如，选用即将被市场淘汰的技术，或是刚刚推出但还不成熟的技术，都会导致信息孤岛的形成。

（3）管理原因。管理方面的问题是信息孤岛形成的最重要原因，而在管理原因中最主要的是缺少信息化战略规划。由于缺少信息化战略规划，致使企业在信息化实施过程中，采取一种“积木”式的策略，每个信息系统都是独立地建设，必然形成许多个信息孤岛。

其次，缺少信息资源规划也是信息孤岛形成的重要原因。不重视信息资源的规划是许多企业共同的薄弱环节。由于不重视信息资源规划，因此，致使信息系统在开发和应用之初就先天不足，一个信息系统本身信息资源管理都处于混乱状态，更遑论信息交换和共享了。

（4）业务流程原因。在一些企业，手工处理的业务本身就存在许多弊端，如，由于业务流程被人为地分割成若干孤立的业务项，而不同的业务项分别由不同的业务人员负责，而且，这些业务人员之间又缺乏交流，形成了许多“业务孤岛”。显然，这些“业务孤岛”的存在就决定了再好的信息系统也无法避免信息孤岛的出现。

（5）标准的原因。企业在不同时期分散实施的系统，由于在运行环境、数据库系统、信息编码规则、业务流程定义等方面执行不同的标准，导致无法实现信息共享和系统集成。例如，某企业开发了人事管理信息系统，在信息编码上，标准不统一，有的用员工编号，有的用身份证号，也有用自然序号或姓名，从而为系统埋下隐患。

12.3.2　信息孤岛的预防及应对

信息孤岛问题应从两个方面解决，一是消解已有的信息孤岛，二是预防新的信息孤岛的形成。在企业信息化实施过程中，消解已有的信息孤岛和预防新的信息孤岛，二者

不可偏废。如果只重视预防而忽视消解，必然造成新旧信息孤岛之间存在许多“死结”，使得新的信息系统由于受到旧系统的拖累而无法正常运行；反之，就会造成已有的信息孤岛还没解决，就又出现新的信息孤岛。

在实施过程中，如果实际情况发生改变，首先的工作是修改规划，然后，再按规划去执行，而不是把规划放在一边，按主观意愿去实施。

1．消解已有信息孤岛的措施

（1）升级替换。升级替换就是对现有失去持续维护能力和没有维护意义的系统，采用升级的办法或用全新的系统替换，将旧系统中产生的数据导入到新系统中，从而消除现有信息孤岛。在升级替换过程中，应当把系统运行环境、数据库系统等统一起来，在数据整合的基础上实现系统的整合和业务的集成。条件许可的情况下，可以对企业信息化重新规划，对现有运行的系统进行全面的升级和替换。

（2）建立数据交换协议和数据接口。建立数据接口是一种不彻底但有效的方法。我们知道，企业信息化是一个渐进的过程，因此，在大多数企业里，都有先后开发的相对独立的信息系统，并且这些系统可能由不同的供应商提供，同时，由于条件限制，不可能做到全面升级。这时，可以采用建立数据接口方式实现系统之间的数据交换和信息共享。接口协议建立后，各供应商负责自己系统的修改，并按照接口协议规定存取所需要的数据。

（3）通过集成平台实现系统应用的集成。集成平台是实现不同信息系统之间数据交换和信息共享的通用工具，它是在应用集成技术支持下完成的。集成平台将业务流程、公共数据、应用软件、硬件和各种标准联合起来，实现不同信息系统之间的无缝集成，像一个整体一样进行业务处理和实现信息共享。这种集成平台还可推广到跨企业的信息系统的集成，实现多个企业信息系统之间的商务交易。

2．统一规划，预防信息孤岛

一个准备实施的信息化系统，一定要把预防信息孤岛作为重要的任务之一。企业管理层应从预防可能产生信息孤岛的环节入手，进行全面的系统规划、统一接口标准、建立业务流程定义规范，从而保证信息系统具有较强的鲁棒性，保证系统以健康的状态投入运行。

（1）统一信息化规划。首先，要结合企业实际情况，进行信息化规划，而后，在规划的指导下，确定统一的系统运行环境、数据库系统，规定信息集成模式、接口标准和规范。在实施新的信息系统前，依据总体规划要求，审查系统的功能接口，并对可能会出现的问题进行模拟实验，以避免新的信息孤岛的产生，实现各个子系统的高速、高效互联，达到信息共享和网上数据交换，提高信息传递效率。

（2）理顺企业的数据流。在企业信息化实施过程中，数据处理的准确性、及时性和可靠性是至关重要的。它是以各业务环节数据的完整性和准确性为基础的，因此，理顺企业的数据流是企业信息化建设成功的关键之一。信息化建设是对企业管理水平进行量

化的过程，也是提高管理水平的良好手段。企业流程包括管理流程、业务流程和生产流程，无论是哪种流程，都必然有数据流相伴随，因此，理顺了数据流也等于理顺了管理。而理顺数据流就要明确部门之间哪些数据需要共享，哪些数据要上报企业领导，哪些部门需要获取外部的知识或信息，哪些数据需要对外发布和宣传，哪些数据需要保密，子公司与总公司交换哪些数据等等。

（3）统一数据平台和集成标准。统一数据平台的基础就是以业务为核心，通过信息流将企业各部门的主要业务集成起来。信息系统建设，需要根据企业规模确定数据结构，即是采取分布式数据结构还是集中式数据结构。在此基础上，统一数据库系统和运行平台。当采用分布式数据库时，需要确定分布的层次、数据传递方向和标准，这样有助于减少日后信息孤岛现象的发生。

12.3.3 信息资源分类

信息资源分类是根据信息资源自身内容的属性或特征，将其按一定的原则和方法进行区分和归类，并建立起一定的分类体系和排列顺序。根据不同观察角度和需求的不同，可以对同一个信息资源进行多重分类。

信息资源分类的主要功能是用于信息资源的管理，而信息资源管理的起点和基础是建立信息资源目录。信息资源只有科学地建立了目录，才能使信息资源得到快速、及时的存储、处理、检索和使用。

1．信息资源分类的基本原则

信息资源分类是信息资源管理中最为复杂的工作之一，应当遵循以下三个原则：

（1）简洁原则。信息资源的分类必须简洁，类目的设置一般以分到二级类目为宜，至多不超过三级类目。因为超过三级类目后，从管理的角度，分类的难度和工作量将会有很大的增加；同时，从使用者的角度，对信息资源的分类查找变得更加困难，这将偏离对信息资源进行分类的初衷，难以达到对信息资源快速查找定位的目的。

（2）独立原则。在设置不同的分类方式时，不同分类方式的类目设置尽量不要重复，即需要从不同的维度对信息资源进行分类，不同的分类维度最好能够正交。

（3）可操作原则。需要注意分类标准的可操作性，很多信息资源的分类都是基于信息资源的业务分类进行细化和合并，尤其对于细化的情况，除非有明确的判定规则和判定依据，应尽量避免细化分类，大多数情况下，对已有类目进行细分是一件非常困难和无法完成的工作。

2．信息资源分类方法

对信息资源的分类一般从三个维度展开。一是从信息资源管理的角度对信息资源进行分类，二是从信息资源的来源和提供部门的角度进行信息资源的分类，三是从不同的应用主题，建立信息资源的分类体系。

（1）从管理维度分类。从信息资源建立的最初目的来看，一般信息资源都是在业务

信息化基础上形成的各种信息。例如，对自然人管理形成的人口信息资源、对法人管理形成的法人信息资源、对各种自然资源以及空间基本信息进行管理的自然地理与空间信息资源等等。各领域、各部门在信息资源的采集加工过程中，从业务管理的角度一般都有比较明确的信息资源分类，用于指导信息采集的专业分工。同时，管理角度的信息资源分类也是专业业务分工的结果，某些情况下，二者相互影响。

（2）从专业维度分类。专业业务分工决定了管理维度的信息资源分类，而从业务整体角度考虑，在信息和业务整合的过程中，信息资源的管理分类可以促进专业部门更好的合作。从实际操作的情况看，管理维度的分类一般有两种情况，一是专门的业务部门所采用的信息资源管理分类体系。例如，一般地震信息资源的分类包括了测震、前兆、强震动、应急等主要分类。二是综合部门从信息资源登记和管理的角度提出的分类，例如，部分地方政府的信息化管理部门从信息资源管理的角度，已经初步制定了政务信息资源的分类方式。

（3）从信息来源维度分类。这种信息资源的分类体系比较简单，一般按照权威的信息资源提供部门，设置信息资源的一级、二级乃至三级类目。按照信息资源的来源进行分类的最大优势在于两个方面。一是从分类信息的赋值角度极大地简化了工作量，基本上在数据采集的过程中不需要对采集人员进行专业培训，甚至不需要进行重复录入。对同一个信息资源提供部门来说，只要设定一个初始化的数值，其后该单位所有的信息资源分类信息都可以复用。采用信息资源来源分类的第二个优势是使用者不需要学习或者了解特定分类体系的内容，使信息资源的查找过程更加简单和直接。

（4）从应用主题维度分类。根据应用主题对信息资源分类是三种分类体系中最为复杂的一种分类方式。同一个信息资源根据其服务和应用的目标不同，会有不同的分类方式。例如，对全国行政区划信息而言，从基础测绘部门的角度，可按照其服务对象进行分类，一般划分到基础性信息资源中；而对于宏观经济管理而言，该信息资源由于不是核心业务信息，可能被划分到辅助性信息资源中。尤其对于政府部门产生的信息资源，从政府部门之间跨部门应用角度提出的分类体系与服务于企业和公众的信息资源分类体系必然有很大的不同，这是由不同的信息资源使用者的应用需求来决定的，有多少不同的应用需求就有多少相应的信息资源分类方式。

很多情况下，不同信息资源分类体系需要相互转化和映射。当一个信息资源已经采用某种分类体系进行分类后，再按照其他分类体系进行赋值时一般都希望通过分类转换的方式自动进行，而不是重新进行分类赋值。

12.3.4　信息资源管理基础标准

信息资源管理基础标准是指那些决定信息系统质量的，因此也是进行信息资源管理的最基本的标准。信息资源管理基础标准有数据元素标准、信息分类编码标准、用户视图标准、概念数据库标准和逻辑数据库标准。

1．有关原则

为了有效地制定和实施上述标准，应当遵循一些重要的原则：

（1）不能把例外当成正规。任何原则都有例外的情况，没有适用于所有情况的标准。但是，数据管理人员决不允许把例外当成正规。

（2）管理部门必须支持并乐于帮助执行标准。如果违背了标准，管理部门必须帮助确保那些违背标准的行为得以纠正。

（3）标准必须是从实际出发的、有生命力的、切实可行的。标准必须以共同看法为基础，标准中复杂难懂的东西越少就越好执行，要保持标准的简明性。

（4）标准决不是绝对的，必须有某种灵活的余地。尽管有些标准必须严格遵守，但是大多数标准不应该严格到严重束缚数据设计人员灵活性的程度。

（5）标准不应该迁就落后。标准要控制和管理当前和未来的活动，而不是恢复和重演过去的做法。在大多数情况下，今天制定的标准是几个月前数据设计所未曾采用的。

（6）标准必须是容易执行的。要达到这一点，必须容易发现违反标准的情况。能自动检查标准符合情况的方法愈多，标准本身愈加有效。

（7）标准必须加以宣传推广，而不是靠强迫命令。即使上级主管部门完全支持数据管理标准，也要向各级业务人员宣传这些标准。数据管理人员必须热情地向所有职员宣传这些标准，向他们讲明为什么需要这些标准。数据管理标准要求程序员和分析员改变他们的数据设计方法。任何持久的、有意义的变化必须来自员工自己的认识。

（8）关于标准的细节本身并不是重要的，重要的是有某些标准。数据管理人员必须善于综合考虑和商讨所要制定的标准细节。

（9）标准应该逐渐地制定出来，不要企图把所有的数据管理标准一次搞完。一旦标准制定出来，就要开始执行，但执行标准是渐进的、有节奏的。允许非数据管理人员有充足的时间对新的标准做出反应和适应。标准的实现必须是渐进过程，而不是突变过程。

（10）数据管理的最重要的标准是一致性标准，包括数据命名、数据属性、数据设计和数据使用的一致性。

2．信息资源管理标准

信息资源是企业最重要的资源，开发和利用好信息资源是企业信息化管理的出发点，也是企业信息化管理的归宿。建立信息资源管理的基础标准，从而保证标准化、规范化地组织信息，这是开发利用信息资源的基本工作。有的企业重视硬件轻视软件，或者重视软件轻视数据，或者重视信息通信网建设而轻视信息资源网建设，这都是一些只见树木不见森林的做法，是造成一些系统失败的主要原因。信息资源管理有下列 5 个基础标准。

1）数据元素标准

数据元素是最小的不可再分的信息单位，是一类数据的总称，它是数据对象的抽象。研究表明，数据元素具有“原子意义”，根据企业的类型和规模，数据元素不仅在数目上

存在统计规律，而且还有比较稳定的对象集。对数据元素的标准化管理包括数据元素的定义、命名和一致性管理。

由于在不重视数据元素的标准化，即缺少数据元素命名标准，或者缺少考虑数据元素创建和使用规划，从而导致数据处理系统中所使用的大部分数据名称庞杂混乱，少数数据元素有众多的同义词或别名。例如，在数据处理系统中的“职工姓名”、“员工姓名”、“职员姓名”等等。

数据元素标准有可分为数据元素命名标准、数据元素标识标准、数据元素一致性标准。

（1）数据元素命名标准。数据元素命名的原则是用一个简明的词组来描述一个数据元素的意义和用途。例如，职工编号。一个单位可能有几个、几十个，甚至上万个、几万个职工，给每个职工一个编号，“职工编号”就是这个数据集的名称。

（2）数据元素标识标准。数据元素标识即数据元素的编码，是计算机和管理人员共同使用的标识。其作用是便于用户的使用，同时，也便于计算机处理。

（3）数据元素一致性标准。保持数据元素名称和数据元素标识在全企业中保持一致，是非常重要的，否则，就会造成数据的混乱。所谓数据元素的一致性，就是不允许有“同名异义”的数据元素，也不允许有“同义异名”的数据元素。

2）信息分类编码标准

在企业信息化的过程中，标准化越来越重要，特别是信息的标准化，更是信息化的“瓶颈”。有这样的案例：在有的企业，由于没有实现信息标准化就盲目地开发了信息系统，运行一段时间后，由于数据混乱而导致系统无法运行。因此，信息分类编码标准是信息标准中的最基础的标准。

信息分类就是根据信息内容的属性或特征，将信息按一定的原则和方法进行区分和归类，并建立起一定的分类系统和排列顺序，以便管理和使用信息。信息编码就是在信息分类的基础上，将信息编码对象赋于有一定规律性的、易于计算机和人识别与处理的符号。具有分类编码意义的数据元素是最重要的一类数据元素，它们决定着信息的自动化处理、检索和传输的质量与效率。应该遵循国家已经发布的有关标准化文件，按照一定的序列，建立起属于本企业的一整套信息分类编码标准。

3）用户视图标准

用户视图是一些数据元素的集合，它反映了最终用户对数据实体的看法。用户视图是数据在系统外部而不是内部的表象，是系统的输入或输出的媒介或手段。用户视图主要包括企业管理的表单、报表、屏幕数据格式等。

用户视图的规范化管理包括用户视图名称、标识和组成的管理。规范并简化用户视图是企业内外信息共享和交换的基础。因为，人工管理积累了很多冗余或不一致的单证或报表，按照这样的格式建造数据库，必然导致低劣的数据环境。要改变这种情况，必须从用户视图标准化开始设计信息系统。

4）概念数据库标准

概念数据库是最终用户对数据存储的看法，是对用户信息需求的综合概括。用户的信息需求首先集中在概念数据库的界定上。概念数据库标准包括数据库名称、标识、主关键字和数据内容列表。列表项可以是数据元素，也可以是数据元组。

概念数据就是主题数据库的概要信息。企业的概念数据库标准是指列出全企业所有的主题数据库的概要信息，它是经过战略数据规划工作之后完成的。由于企业性质和规模的不同，有的企业可能有 30 个左右，有的可能有 50 个左右的主题数据库概要信息，这就是企业概念数据库标准。

5）逻辑数据库标准

逻辑数据库是信息系统分析设计的基础，是对概念数据库的进一步分解和细化，一个逻辑主题数据库由一组规范化的基本表构成。

由概念数据库演化为逻辑数据库，主要工作是采用数据结构规范化的理论与方法。将每个概念数据库分解、规范化成三范式的一组基本表。企业的逻辑数据库标准是指以基本表为基本单元，列出企业全部的逻辑数据库。

3．数据字典

数据字典是数据管理的重要工具，是有关数据的信息的收集、维护和发布的机制，特别是在制定“信息资源管理基础标准”的过程中，建立数据字典更是一项非常关键的工作。数据字典提供了关于数据元素、元组、记录组、信息的定义和使用机制，以及这些数据实体之间的联系。还可以定义其他一些对象，例如，输入格式、报表、屏幕界面、处理程序模块等。

12.3.5 建立业务概念设计模型

信息资源规划之所以要进行业务分析，是为了按信息工程的思想方法来重新认识企业，以便能系统地、本质地、概括地把握企业的功能结构。在信息工程方法论中，是用“职能域→业务过程→业务活动”这样的层次结构来把握企业功能的，这就是企业业务模型。

1．业务模型设计步骤

业务模型的设计过程可分为三步：第一步，建立业务职能域模型；第二步，识别定义每个职能域的业务过程；第三步，列出每个业务过程的各项业务活动。

建立正确的业务模型，是一项复杂而细致的认识活动。开发人员要与企业高层领导和各级管理人员来共同分析企业的现行业务，按照各种业务的逻辑关系，将它们划分为若干职能区域，弄清楚各职能区域中所包含的全部业务过程，再将各个业务过程分为一些业务活动。

显然，建立业务模型的任务不是一两个人一次完成的，而是多人多次反复完成的。即使是职能岗位的管理人员，也未必能一次讲清楚所涉及的业务过程和活动，因为还需

要一些分析方法和表述方法。因此，进行业务分析建立业务模型的过程，是对现行业务系统再认识的过程。

提出业务模型是企业信息化的基础性工作，因为，信息化最初阶段就是将人工的业务过程和业务活动，变为以计算机为信息存储处理工具的自动化或半自动化的过程和活动。在此基础之上，使用信息技术改造原有手工业务模式，使原来的业务过程和活动发生某些根本性的变化。因此，首先要搞清楚现行系统的业务过程和活动，然后考虑使用信息技术对这些过程和活动带来必然的调整和改造，这就是业务模型分析工作的实质。

2．设计职能域模型

职能域（职能范围、业务范围）是指一个企业或组织中的一些主要业务活动领域，例如，工程、市场、生产、科研、销售等。一个中型制造厂的职能域有经营计划、财务、产品计划、材料、生产计划、生产、销售、配送、会计、人事等。

定义职能域是信息资源规划第一阶段的一项重要任务。由于职能域是企业功能划分的抽象，不是现行机构部门名称的翻版，因此，一经识别定义，就要保持相对的稳定性。这项工作应该在规划工作一开始就尽快着手进行，应该在企业高层管理人员的参与下，由规划核心小组完成。职能域的列表也就是职能域模型，可以用如上形式的列表，也可以用方框图表示。

当一个企业的职能域划分出来以后，就可以进一步明确信息资源、规划的范围或边界。设计职能域模型时需要特别注意的一点，是控制把握好规划职能域的数目。尽管要面向全企业，但不是无所不包地定义大量的职能域；可以包括大部分或主要部分，但不是一两个职能域。如果规划的范围太广而没有抓住重点，可能造成无法控制的局面；如果范围太窄，那么自顶向下规划的作用就显示不出来。只有包括一定数量的职能域，大量的信息资源相互交叉作用才能反映出来，从而使规划工作能为信息系统提供坚实的基础。

企业最高层领导团队应认真审核规划出来的职能域模型是否完善，并明确自顶向下规划的范围。

3．设计业务过程模型

每个职能域都包括一定数目的业务过程，例如，一个中型制造厂有十几个职能域有 30 多个业务过程；一般大型企业大约有 30 个职能域，150～300 个业务过程。

（1）业务过程的命名和定义。业务过程的命名应符合它们所起的功能；一个业务过程可以用简单的短文来加以定义。例如，对“库存管理”的业务过程可以定义为“对仓库的原材料、零部件进行接收、保管、发放并核算库存量”。业务模型的第二层结构是业务过程的识别、命名和定义。这项工作主要依靠用户分析员来完成，当然他应该同其他业务人员商讨，使命名和定义有一致的理解。

（2）业务过程与组织机构。职能域和业务过程的确定，应该独立于当前的组织机构，因此，为强调这一点，应称为“逻辑职能域”。组织机构可能变化，但企业仍然会执行同样的职能和过程。职能域与业务过程的确定，主要应该考虑独立于当前组织机构的职能，

因此会有这样两种情况：

- 经逻辑分析而得出的职能模型中可能包括这样的职能域，它横跨两个或多个现行系统的业务部门。
- 对现行系统所列出的业务过程中可能会有这样的一些过程，它们分别属于不同的职能域，但功能相同或相近。

例如，“库存管理”过程，可以由一个独立部门也可以由几个部门来完成，尽管仓库职能部门是时而分开，时而合并的，但库存管理的业务过程总是在起作用。

4．识别业务过程的参考模式

识别业务过程一般来说缺乏较好的形式化方法，主要靠有经验的业务人员和分析人员进行反复提炼。但是，可以提出一种参考模式，帮助设计师发现并列出其业务过程。这种参考模式就是“三类型四阶段生命周期模式”。“三类型”是指产品型、服务型或资源型三种类型，任何企业都可归入这三种类型。“四周期”是指任何类型的企业的业务过程都大致分为 4 个阶段。其中，产品型企业参考业务过程如下。

（1）计划阶段：需求调查、市场研究、设计、生产能力计划、核算等。

（2）获得阶段：采购原材料、补充人员、生产调度、加工制造、检测等。

（3）保管阶段：成品入库、库存管理、质量管理、包装等。

（4）处置阶段：交货、销售、订货服务、发运、付款、废品处理等。

服务型和资源型企业或组织，也有类似的四阶段生命周期。

5．业务活动分析

（1）业务活动分析步骤。在每个业务过程中都包含一定数目的业务活动。业务活动是企业功能分解后最基本的、不可再分解的最小功能单元。对业务活动命名可采用一个动词，以表示该活动所执行的操作。一般每个业务过程含有 5～30 个业务活动。在一个小型的企业里可能有几百个业务活动，而在一个大型复杂的企业中可能有几千个业务活动。在做业务分析时，一般是把职能域分解成多个功能，每个功能再分解成更低层的功能，这样逐级向下分解，直到产生最基本的活动为止。

（2）凝聚性活动。有专家在总结活动分析工作的基础上，提出活动模型中每一活动应是凝聚性活动，为寻求这样的活动，列出的特征如下：

- 一个凝聚性活动产生某种清晰可识别的结果。这种结果可以是销售一件产品、一个想法、一个决策、一组方案、一份工资单、一次顾客服务等等，应该能用一个简单的句子来说明这个活动的目的或结果。相比之下，一个非凝聚性的活动，总是产生不可确定的结果，或者几个无关的结果。
- 一个凝聚性的活动有清楚的时空界限。在这个确定的时间和空间里，可清楚地指出，谁在这个活动中工作和谁不在这个活动中工作。凝聚性的活动之间的转换具有清楚的标志，而非凝聚性的活动则互相重叠混杂，不能确定在何时何地进行。
- 一个凝聚性的活动是一个执行单元。它明确规定一个人或一个小组去产生结果，活动的管理职责也有类似的明确规定，由一个人或一组人负责。

- 一个凝聚性的活动在很大程度上是独立于其他活动的。

凝聚性活动都是基本活动，如果发现有的活动不具备凝聚性特征，就要进行调整；有的活动是重复的或者相当接近的，就要清除重复活动的多余部分，合并相似的活动，才能得出良好的业务活动模型。

6．业务模型的复查与确认

经过对业务活动的分析以及识别所有的凝聚性活动，再按相互联系的紧密程度分组，就可以积聚成一些业务过程。对业务过程再组合，就可以形成若干个逻辑职能域，以作为共享数据库的信息系统基础。就是说，逻辑职能域是对按企业的职能部门划分的职能域的修正。同样，按业务活动分析组合起来的业务过程模型，是对按业务人员的经验初步建立起来的业务过程模型的修正。

当初步的企业模型以图表形式得出以后，要进行认真的复查和审核。复查工作要在核心小组的组织下，除充分发挥用户分析员的作用外，还要有层次地与管理人员对话，请他们进行仔细的审查。

审查中应注意以下三点：

（1）完整性。这种模型应该是表示组成一个企业的各个职能域、各种过程和活动的完整图表。

（2）适用性。这种模型应该是理解一个企业的合理有效的方法。在每一个分析层次上职能和活动的确定，对于参与工作的管理人员来说都应该是觉得自然的和正确的。

（3）永久性。只要企业的目标保持不变，这种模型就应该保持是正确的和有效的。

12.4　企业信息化实施

信息化实施是一个长期渐进、不断优化、逐步完善的过程，并没有统一、固定的模式，而是要从企业实际情况出发，根据企业业务发展的实际需要，制定切实可行的信息化实施计划，并通过坚持不懈的努力，逐渐找到一条符合企业发展需要的，行之有效的企业信息化发展道路。

12.4.1　信息化实施过程

一般来说，企业信息化的实施过程主要应从如下几个方面着手。

1．确定信息化总体思路

任何一个企业，不论规模大小、实力强弱，在实施信息化前，要制定信息化总体思路，并注意以下几点。

（1）必须对信息化有一个总体的、综合的、相对长期的考虑，要认识到，信息化是一个综合性、系统性和整体性的系统工程，涉及到企业的方方面面，因此，不能为信息化而信息化。

（2）信息化的发展思路：整体规划、分步实施；效益驱动、突出重点；由易到难、逐步完善。

（3）具体实施时，要采取在总体规划的基础上，分阶段完成，由局部到整体，逐步扩大应用规模，使企业信息化得到有计划、有步骤地推进。

（4）明确不同阶段的实施重点以及不同部门在企业信息化过程中的不同职能，做到“统一规划、统一投资、统一标准、统一建设、统一管理”，保证企业信息化的健康发展和有效实施。

2．制定信息化实施规划

信息化实施规划主要包括以下几个方面的内容：

（1）明确发展目标和实施重点。企业要从自身的实际需要和现实条件出发，确定企业近期、中期以及长期的企业信息化的任务和要求，做到方向正确，目标清晰，实施进程有保障。

一个企业信息化在不同的阶段应有不同的重点，只有重点突破，实施工作才能事半功倍。例如，一个加工制造企业，要上 ERP 项目，那么，在半年到一年的时间里，ERP 肯定是实施的重点，其他与之关系不大的项目都可适当延后。

（2）成立领导机构。企业信息化是一个长期性、综合性的系统工程，牵涉面广、实施难度大，如果没有企业高层领导的支持和推动，没有有效的协调与管理是很难取得预期效果的。所以，成立一个由高层领导挂帅、有关部门领导参加的领导小组或项目委员会十分必要。信息化领导小组组长或项目委员会主任最好由企业“一把手”担任，CIO 担任副职，领导机构办公室一般设在 IT 职能部门。

（3）做好企业业务信息化需求分析。企业所有的信息化项目都不是孤立的，无一例外的都是为业务服务。因此，要做好业务的信息化需求分析十分重要，它是信息化成功的必要条件。需要指出的是，这类需求包括现实需求和潜在需求，而以潜在需求分析最为关键。

（4）确定企业信息化不同发展阶段的投资预算。信息化建设必然需要一定的资金以及其他资源的投入，在制定信息化实施规划时要充分考虑到，并做出科学合理的安排。在安排企业信息化预算时，既要避免不切实际的盲目投入，造成无谓浪费，另外，也要满足基本需求，以免导致“烂尾工程”和“半截子工程”的出现。

（5）制定必要的促进企业信息化发展的规章制度。企业信息化牵涉面广，实施难度大，制定必要的规章制度，使企业信息化发展有章可循，有据可查，有标准可执行，对加快信息化的顺利发展有重要的促进作用。此外，还应在企业信息化实施规划中明确信息化实施效果评估的方法，信息化方案优化等措施，以便在信息化发展中不断总结经验，保证信息化发展的实际效果。

3．制定企业信息化的实施策略

在制定信息化实施策略的过程中，应该遵循以下原则：

（1）效益原则。尽管信息化的效益很难确切衡量，但是它会在企业的运营过程中体现出来，要全力避免为信息化而信息化的“形象工程”、“面子工程”，必须坚持效益原则。

（2）实用原则。企业实施信息化时要根据自身的业务特点，选择实用的信息化实施方案，要解决实际问题、产生实际效果，使企业信息化落到实处。

（3）系统性原则。不同规模和基础的企业在实施信息化时各有不同的侧重点，但在实施过程中都应考虑到企业的整体发展需要，以更好地实现与其他业务部门的应用集成，尽量避免在信息化过程中产生新的信息孤岛，影响企业信息化的实际效果。

（4）可扩展性原则。企业信息化发展是一个长期的过程，也是一个不断优化，逐步完善的过程，所以，企业的信息化建设应考虑到扩展性问题，在硬件的配置、软件的开发、系统的建设等各方面都应注意到企业今后发展的需要，使企业信息化建设能有序推进。

由于企业信息化实施需要调动各方资源，牵涉到方方面面，因此，制定企业信息化实施策略很有必要，应注意以下几点：

（1）要从基础到高端。企业信息化应该首先解决基础管理方面的问题，然后再解决更高层次的问题。选择技术和设备也应从现实出发，强调实用性，而不要过分追求技术和产品的高档次，要把先进的技术和自身的应用实际统一起来。

（2）要从单项到集成。

（3）要从内到外。企业信息化要从理顺内部的业务流程出发，逐步延伸到企业外部，把企业对客户、供应商的管理纳入到信息化发展的范畴，逐步构筑起完善的客户关系管理、供应链管理等系统，使企业的信息化发展跃上一个新的台阶。

（4）先易后难。一个企业的不同部门、不同系统、不同业务在实施信息化过程中的难易程度是不同的，应先实施难度小的项目，最后再实施难度大的项目。

4．夯实企业信息化的基础工作

坚实的基础是信息化成功的必要保证，企业应在信息化实施过程中做好以下几项基础工作：

（1）逐步完善信息化的基础设施。企业信息化必须依靠一定的基础设施。因此，企业必须做好信息化基础设施建设，包括从硬件、通信和网络设施，到软件平台和数据平台的建设。

（2）不断提高全员信息化素质与技能。企业在实施信息化过程中，应特别注意提高全员，特别是主要业务人员的信息化素质和技能。要做好不同层次人员的培训。

（3）营造良好的企业信息化环境。良好的企业信息化环境对企业信息化实施具有很大的促进作用。因此，企业的领导和管理人员一定要成为企业信息化的倡导者、实践者和受益者，从而影响和带领企业员工更加自觉、主动地参与企业信息化建设。

（4）抓好企业信息化的基础管理工作。数据是信息化的最基本原材料，因此，做好数据资源规划和管理是信息化的基础工作，它能够有助于信息化的顺利实施。

5．做好信息化阶段性评估

企业信息化实施是一个复杂的过程，往往尽管有系统的规划和比较周密的计划，但是，真正到了实施阶段，意外的情况和问题还是会经常出现。因此，不断地进行信息化阶段性评估是十分必要的。评估中要注意以下几方面问题：

（1）实施过程与规划的吻合度。在信息化实施过程中，有一个比较普遍的现象，就是随着实施的进展，其与原有规划的背离程度会不断加大，到最后，已经变得完全两回事了。之所以发生这种现象，其主要原因就是缺少阶段性评估。通过评估，不断发现问题，并根据新的问题修正信息化规划，使之成为整个信息化实施的指导纲领。

信息化阶段性评估还有一个重要内容，就是风险评估。由于信息化的实施，可能引发业务流程，乃至组织结构的改变，同时，不可避免地引起利益的变化，甚至利益冲突。而这些问题，可能潜伏着某种风险。因此，企业应在信息化实施的过程中，不断收集可能的风险因素，发现风险苗头，并对风险进行评估，以求规避风险。

（2）做好各利益相关方的沟通和协调。大多数企业（甲方）信息化项目都是委托给专门机构（乙方），有些企业还专门聘请监理机构（丙方）对项目实施监理。这样就形成甲乙丙三方的关系。有时，乙方只是承包方，其后面还有软硬件供应商。

在这几方利益交织的情况下，如何使得各利益相关方能够齐心协力，为信息化实施做好各自的工作，往往是一个很具挑战的课题。这时，甲方毫无疑问是主导者，必须与各方做好沟通和协调。

（3）做好新老业务模式的衔接和转换。信息化实施后，必然形成新的业务模式，例如，会计电算化后，手工记账的业务模式就要变成电子记账的业务模式。如何实现新老两种业务模式无缝的衔接和平滑的转换，是信息化实施中具有攻坚意义的任务。要做好这项工作，一是从信息化规划开始，就要有业务人员参加。这些业务人员通过参与信息化实施的全过程，早已对新的业务模式了然于胸，因此，在业务模式转换过程中，就会驾轻就熟；二是信息化项目应遵循“用户友好原则”，为其用户提供友好的界面和平台；三是做好培训。

（4）做好信息化的系统优化。企业信息化的过程是一个不断完善的过程，也是一个不断优化的过程。

（5）做好信息化的横向和纵向扩展。当企业信息化在某些领域或某些部门实施成功后，就应当考虑扩展，逐步扩大应用范围，加深应用深度，从而，不断提高信息化水平。

12.4.2 业务流程重组

1990 年，Michael Hammer 博士首先提出了业务流程重组（Business Process Reengineering，BPR）的概念。由于 BPR 理论突破了传统的企业分工思想，强调以流程为核心，改变了原有以职能为基础的管理模式，为企业经营管理提出了一个全新的思路。

1．BPR 的概念

Hammer 认为，BPR 是对企业流程进行根本反思，要对其进行重新设计，从而使得衡量现代企业绩效的关键指标，例如，成本、质量、服务和速度等得到奇迹般的改善。

Hammer 对 BPR 的定义较全面地反映了 BPR 的本质特征，这就是以企业流程为核心、对企业流程进行根本反思、彻底重新设计企业流程，使企业发生跨越式的发展。

以往的企业管理的变革和改进都是基于传统的职能分工理论，因此，其效果往往不佳，造成“膨胀→精简→再膨胀→再精简”的恶性循环。而 BPR 理论从根本上打破了职能分工理论的局限，把企业流程作为基础和核心，作为管理变革的对象。

所谓企业流程是指为了完成某一目标或任务而进行的一系列跨越时空的逻辑相关活动的有序集合。通过考察企业流程的发生、发展和终结，确定、描述、分析、分解整个企业流程，重构与企业流程相匹配的企业运行机制和组织机构，实现对企业全流程的有效管理和控制，能够使企业真正着眼于流程的结果，消除传统管理中只注重某一环节而无人负责全流程的弊端。

2．BPR 的内容

BPR 彻底地抛弃了传统的职能管理模式，对员工角色、管理观念、组织机构带来巨大变革。BPR 强调 4 个核心内容，即“根本性”、“彻底性”、“戏剧性”和“流程”。

（1）根本性。BPR 强调要进行根本性的再思考，各方面都要关注“流程”，因为，它是企业的核心问题。为了使得思考有方向和目标，要提出一些问题。例如，我们为什么要做现在的工作？我们为什么要用现在的方式完成这项工作？为什么必须由我们而不是由别人来做这项工作？通过对这些企业运营中最根本性的问题的思考，就会发现以前视而不见的问题。

（2）彻底性。彻底性是要求对 BPR 进行追根溯源，对既定存在的事物不是进行小修小补，而是要进行彻底的改造。例如，BPR 彻底改变了员工的地位，员工不再是被动的命令执行者，而是被赋予足够的权力和负有与之匹配的责任的流程主人。BPR 鼓励员工在权力范围内自己决策，高层管理人员只是进行必要的指导和协调。同时，BPR 对员工的素质要求更高了，要求员工能胜任多层次的工作。在工作开始阶段，BPR 更强调观念教育而非技能培训。

（3）戏剧性。戏剧性表明 BPR 完全抛弃传统管理观念，不是追求稍有改善，而是充分强调结果的满意度；在衡量员工的业绩标准上，BPR 注重员工创造的价值，倡导创新，这与传统的“多劳多得”有本质的不同；BPR 彻底否定企业中大量的低效劳动和无效监督控制；另外，员工的晋升是靠综合能力而非一时的工作业绩。

（4）流程。BPR 主张不是企业的业务流程的简单改善，而是要创建全新的组织机构，打破以专业分工理论为基础的职能部门管理框架，建立以流程工作小组为单元的管理模式，形成扁平式管理机构，大大压缩了管理层级，不但提高了管理效率，增强组织柔性，而且节约了中间管理层所产生的巨额成本。

3．BPR 的作用

BPR 的实施将使企业发生根本性的变革，增强企业的活力，给企业带来巨大的经济效益。

（1）BPR 的实施使企业更贴近市场。企业为了提高顾客满意度，将主动进行市场调研，预测市场需求，及时掌握市场走向。同时，管理层级的压缩，使高层管理人员与第一线业务人员和顾客之间缩短了距离，能够直接获取一线人员的意见，在第一时间感知顾客对产品的反应和新的需求，从而及时调整经营决策。

（2）BPR 使生产成本成倍压缩。BPR 吸收了先进的管理理论和技术，利用并行工程等思想，可大幅度压缩产品的开发周期，加快产品的更新换代频率；同时，BPR 以企业流程为核心，彻底消除了传统管理模式中人为因素的干扰，减少了中间环节，降低协调、控制成本。另外，BPR 否定了传统管理模式中的多余监控，从而减少管理层级，使得管理成本大大降低。

（3）BPR 使产品质量得到全面提升。BPR 将全面质量管理思想贯穿于整个流程中，从市场调研阶段开始就把产品质量作为重要指标来监控。BPR 考虑了市场反馈信息的作用，采用柔性制造系统等先进理论开发、生产产品，可以最大限度地保证产品质量的全面提升。

（4）服务质量更趋完美。由于 BPR 彻底抛弃了职能分工的思想，确立了以流程为核心的观念，因此，企业所有员工都把满足顾客最大需求作为自己工作的首要目标。在工作方式上，员工由被动服务变为主动服务，传统管理模式下企业管理人员的许多监管工作已变得多余，员工工作的主动性和自觉性大大提高，企业的整体服务水平上升了一个层次，服务质量更趋完美。

希赛教育专家提示：由于 BPR 是企业管理的一场革命，通过根本性的变革建立以企业流程为核心的运营机制和组织机构，可能会给企业带来巨大的震荡。所以，企业实施 BPR 有可能获得巨大效益的同时，也承受了较大的风险，一旦 BPR 失败，那将会给企业带来难以挽回的后果。

4．BPR 遵循的原则

BPR 在追求顾客满意度和员工追求自我价值实现的流程中带来降低成本的结果，从而达到效率和效益改善的目的。BPR 在注重结果的同时，更注重流程的实现，并非以短期利润最大化为追求目标，而是追求企业能够持续发展的能力，而为达此目标必须坚持以流程为中心的原则、团队式管理原则（以人为本的原则）和以顾客为导向的原则。

（1）以流程为中心的原则。企业业务流程，特别是关键业务流程总是在最大程度上体现了企业的总体目标和用户价值，因此，流程式管理模式最主要的特点是企业的一切工作都是围绕结果而不是围绕工序或分工。因此，BPR 注重的是业务流程整体最优，通过理顺和优化业务流程，使得业务流程中每一个环节上的活动尽可能实现最大化增值，尽可能减少无效的或不增值的活动，并从整体最优的目标出发，设计和优化业务流程中

的各项活动，消除本位主义和利益分散主义。

这里有一个典型的例子：传统企业里的销售人员从市场上或顾客那里得到新的产品需求后，将其交给研发部门，然后就只能等待，既不能对开发工作做日程上的监督，也不能对开发中的问题提出建议。然而，他们是企业中对这件事最清楚也最关心的人，因为结果决定着他们的销售业绩。显然，这是一个糟糕的业务流程，但人们已习以为常。而按照以流程为中心的原则，应使销售和研发形成一个完整的业务流程。

（2）团队管理原则。在流程式管理模式下，企业的组织结构必须服从业务流程，要使组织扁平化，而要做到这些，就必须坚持另一个重要原则——团队式管理原则。在 BPR 中，首先是设计业务流程。而后依据业务流程建立或改造企业组织，尽量消除或弱化“中间层”。这不仅降低了管理成本和成本，更重要的是提高了组织的运转效率及对市场的反应速度。

员工素质的提高是 BPR 取得成功的前提条件。在以流程为中心的管理模式下，员工的积极性和主动性必然高于以往，这是因为他们不再满足从事单调、简单的工作，而是承担一定的责任，有一定的权力，在工作中能充分发挥自我，有成就感。而要使员工的积极性和主动性能够得以长期保持，最有效的途径就是组成团队。

（3）以客户为导向的原则。BPR 理论出现于 20 世纪 90 年代，这是与世界经济的发展，社会环境的变化，科学技术的进步，新技术、新方法的推广应用特别重要的是信息技术的迅速发展所分不开的，而信息技术的发展才能保证顾客导向原则贯彻到底。因此，利用信息技术能够有效地帮助企业使得 BPR 得以很好地实施，例如，利用建模的信息工具可以重新设计经营流程；采用计算机网络、数据库和多媒体等技术建立的信息网络，能够加快信息传递，实现信息共享，其结果是将传统的串行工作方式变为并行工作方式，将企业组织结构由垂直型变为水平型，使企业成为协同工作的组织，使得企业的业务流程，特别是关键业务流程与市场接通，与顾客接通。

另一方面，科学技术的发展和管理模式日臻完善，为 BPR 创造了条件。例如，在加工制造行业，柔性制造系统是一种能高效率、高质量地进行多品种、中小批量生产的自动化加工系统，利用它，企业可以快速响应市场变化，满足顾客多样化和个性化需求。一些现代管理模式，如精密生产、准时制造和全面质量管理等，提倡以顾客为中心，以及坚持增值第一和质量第一的理念，都体现了顾客导向的原则。

12.5　管理咨询

管理咨询是一种以智力和知识为企业等组织提供服务的行业。在我国实行改革开放政策伊始，便开始学习和引进国外的管理咨询理论、方法和经验，并结合我国国情开展了管理咨询工作。经过 30 多年的实践与探索，我国的管理咨询事业从无到有，从小到大，逐步成为第三产业的重要组成部分。它在发展经济、优化资源配置、创造和谐有序的市

场环境、提升我国企业核心竞争力等方面发挥了不可替代的作用。

12.5.1 管理咨询概述

根据英国管理咨询研究所的定义，管理咨询是“由独立的、合格的个人或多数人在鉴别调查关于政策、机构、程序和方法中所提供的一项服务工作，他们提出采取适当的行动的建议，并协助执行这些建议。”

根据美国哈佛《企业管理百科全书》的定义，管理咨询是“对现营的事业实行确实的诊断，进而针对经营环境之变化，确立现行事业的基本方针与有关未来的事业结构的方针，然后根据方针来厘定计划并切实执行。”

根据日本《经营学辞典》的定义，管理咨询是指“调查企业的实际经营状态，诊断经营方面的问题，提出具体的改善建议，或者在此基础上对改善建议的落实给予指导”。

管理咨询在 20 世纪 80 年代初导入中国后，通过不断的学习和借鉴日本和欧美国家企业管理咨询的理论、方法和经验，并结合我国国情进行创造性的开拓和发展，逐步形成了具有中国特色的管理咨询的理论和方法。

《企业管理咨询理论与方法新论》给管理咨询下的定义为：管理咨询是由具有咨询资格的专家，应企业的要求，为之提供各种知识和服务，或深入企业现场，运用各种科学方法，找出企业存在的主要问题，进行综合分析，查明问题的根源，提出切实可行的改善方案，进而指导实施，以改进企业经营，谋求可持续发展的一门系统科学。

应当指出，这个定义也存在一些缺陷：一是咨询对象仅限于企业，其实，管理咨询的对象应是以企业为主的各种组织；二是咨询主体仅限于专家，其实，随着管理咨询事业的发展，咨询主体一般是以机构为主；三是咨询机构是接受客户的委托，是一种商业行为。

管理咨询具有中立性、局外性、综合评价性、建议性和指导性。

（1）中立性。管理咨询的第一个特性就是中立性，该特性在外部人员咨询上表现得尤为突出。利用组织外部的咨询机构或咨询师对组织存在的问题进行实地调查分析，这样接受咨询的组织易于得到更加客观公正的建议，一些组织内部不好解决的问题，通常站在中立和局外的立场上的咨询人员就易于解决。

（2）局外性。企业聘请的外部咨询机构或咨询师通常与企业没有利害关系。他们能站在局外人的立场上对企业运营过程中存在的问题做客观分析，进而能较客观提出问题的解决方案。局外性也保证了管理咨询结果的科学性和准确性。

（3）综合评价性。管理咨询是一个系统工程，既要对企业外部经营环境进行分析，又要对企业内部经营条件有系统、全面的了解，在此基础之上进行综合评价，找出存在的问题，并提供帮助实施解决方案。因此，管理咨询的结果都是综合评价的结果。

（4）建议性。咨询机构或咨询师根据对客户进行实地调查获得的一手资料对其的运

营状况进行分析，进而提出改善其经营管理的方案。咨询师只是充当一个辅助者的角色，为客户提出建议性的而非强制实施的改进方案。至于是否采纳，采纳多少，完全由客户自己决定。

（5）指导性。管理咨询并不停留在仅仅为客户提供建议的层面上，更重要的是，咨询机构或咨询师对客户的相关人员进行培训，帮助指导实施改进方案，从而提高组织的运营水平。

12.5.2　管理咨询的类型

从不同的角度，管理咨询可以分为不同的类型。

1．按范围分类

根据范围，管理咨询可以分为全局性咨询与单元性咨询。

（1）全局性咨询。全局性咨询是针对企业的总体情况或企业高层的需求进行咨询，如企业经营总体评价、企业发展战略、市场环境分析等。

（2）单元性咨询。单元性咨询是对企业的某一职能部门或某一下属单位的咨询，例如，企业营销、生产管理、产品开发设计、质量管理、财务管理等部门，以及某一分公司、生产车间等。

2．按咨询层次分类

管理咨询可分为不同层次：战略咨询、业务咨询和专项咨询。

（1）战略咨询。战略咨询业是咨询产业中的最高层次，主要是为委托人提供战略设计、竞争策略、市场分析、业务规划等服务，同时也有一些咨询公司主要面向政府提供政策决策咨询。提供战略与决策咨询服务，对咨询服务机构要求较高，因为，这种咨询意见短时间内难见成效，但是，对于大中型企业，特别是大型跨国企业来说，一个好的咨询意见，其价值极其巨大，可以使之沿着正确的目标取得成功。

（2）业务咨询。业务咨询是对企业管理的各个业务层面的咨询。业务层面通常划分成若干领域，例如，投融资、财务会计、税务、市场营销、人力资源、生产管理、工程技术、业务流程重组、信息化等。

（3）专项咨询。专项咨询也称专题咨询，它一般以项目形式运作。专项咨询业务范围包括，工程项目、管理项目和研究项目等。工程项目的管理咨询主要是市场分析、技术经济分析，以及项目管理分析等；管理项目咨询主要涉及企业运营的专题研究，包括企业组织架构、流程、制度、人力资源等的研究；专题研究项目主要是企业做出重大战略决策前所需的专门课题研究，例如，企业分拆、合并，以及内部重组等，都需要一些支持性的研究。专项咨询与一般的管理咨询是有区别的，前者一般只要求提供思路性的建议，不涉及具体实施。

3．按专业分类

管理咨询从专业的角度，可分为多种类型：财务会计、组织设计、制度设计、流程设计、市场营销、生产管理、质量管理、薪酬绩效管理、人力资源管理与开发、企业文化等专业。

随着市场竞争的加剧，使得管理咨询机构在专业上出现分化，并分别向着两个相反的方向发展，一是专业化，二是综合化。一些大的咨询公司，不断向综合方向发展，相反，大部分咨询公司则向专业化方向发展。一些规模较小的咨询公司，往往只做一个非常细小的领域。例如，CSAI 顾问团只做软件项目范围内的管理咨询。

4．按咨询机构性质分类

根据咨询机构性质的不同，管理咨询分为内部机构咨询和外部机构咨询。

内部机构咨询一般发生在大型或特大型企业。这些企业机关有众多部门，下属有众多的子公司、分公司。在这样的企业里，往往设有专门从事管理咨询的部门或子公司、分公司。这些管理咨询机构，一般既对内，也对外服务；有的只对内不对外。

内部机构咨询具有成本低、时间安排随意、熟悉情况等优点；但其最大的缺点就是对问题往往习以为常，视而不见，不易发现问题，同时，还可能发生因为利益的关系而失去公正性和客观性，从而影响咨询效果。

外部咨询机构，即市场上的咨询机构，其构成比较复杂。有专职从事管理咨询的机构，也有兼职从事管理咨询的机构，它们大部分是高校或科研机构，在完成教学或科研之余，从事对外咨询。

外部机构咨询的优点是客观公正，易于发现问题；其缺点是成本较高，咨询时间需协商，需要较大的时间和人力熟悉企业情况。

希赛教育专家提示：随着市场化的深入，内部咨询机构越来越少，咨询机构的主体为外部咨询机构。

5．其他分类

管理咨询还可以从其他角度进行分类。下面仅举二例：

（1）自我咨询、利害关系者咨询和第三方咨询。自我咨询是指企业组织内部人员进行的咨询活动。其优点有：一是保密性好，保证企业秘密不外泄；二是机动灵活；三是可以实现 PDCA 循环；四是节省开支。同时，自我咨询通常可以作为引入外部咨询机构咨询的准备。

利害关系者咨询，通常是指企业对子公司、注资企业、外协单位、供货商、销售代理商等的咨询；另外，银行或其他信贷机构对借贷企业的咨询亦属此类。

第三方咨询（中介咨询）是指甲方聘请具有某种特定资质的咨询机构（丙方）对乙方企业的咨询，以求公允。

（2）认定咨询。认定咨询是按照某种标准或法规所进行的有特定目的的咨询，即认

证企业是否具有某种资格条件，例如，ISO 9000 咨询、CMMI 咨询等。

12.6　知识管理

知识管理是信息化时代重要的管理理论和管理方法，管理大师彼得·德鲁克认为，21 世纪的组织，最有价值的资产是组织内的知识工作者和他们的生产力。而知识工作者就是最有生命力的资产，组织和个人的最重要任务就是对知识进行管理。知识管理将使组织和个人具有更强的竞争实力，并做出更好地决策。

20 世纪 80 年代，斯坦福大学的保罗·罗默教授曾提出了经济增长四要素理论，其核心思想是把知识作为经济增长最重要的要素，他认为：首先，知识能提高收益；其次，知识需要投资；第三，知识与投资存在良性循环关系，投资促进知识，知识促进投资。在信息化过程中，企业中最大的资产，就是继资本、劳动之后脱颖而出的第三资源，即知识资源。

由于知识是企业最重要的战略性资源，因此，知识管理就成为企业的一个重要的战略任务。在国际化和信息化的大背景下，企业要在激烈的市场竞争中胜出，就要运用集体的智慧提高应变能力和创新能力，即必须有创造和运用知识的能力，同时，也为企业实现显性知识和隐性知识共享提供的新途径。

人在获取知识的过程中，总是离不开信息，总是在与信息打交道，因此，知识管理的过程也是对信息资源的管理过程。毫无疑问，知识管理应是以人为中心，以信息为基础，以知识创新为目标，将知识看作是一种可开发资源。简单说，知识管理就是人在企业管理中对其集体的知识与技能的获得与运用的过程。

12.6.1　知识管理对组织信息化的意义

在全球化和信息化的时代，决定企业成功的最主要的因素就是企业的知识管理能力和知识创新能力。但是，知识管理决不是空中的楼阁，它决不能脱离企业的基础，这个基础就是信息化技术。反过来，知识管理能够促进企业信息化。

1．知识管理是信息化发展的高级阶段

知识管理从结构上可分为对人的管理和对信息的管理两个方面，而对信息的管理又可分为三个层面，最底层的是通信网络，用来支持信息的传播；第二层是高性能计算机服务器层，这是存取信息、数据的关键环节；第三层是信息库、数据库系统层，它是信息管理系统的关键层。它和计算机服务器层一起组成了管理信息系统，为各种信息转化为知识的应用提供了有力的支持。

不难看出，信息库、数据库系统是信息管理的核心。它的功能是：第一，存放经过整理、归类的信息；第二，提供获取各种人类专家的个人经验的工具；第三，为知识更新提供必要的维护手段。其目标就是最大限度地实现知识资源的共享和交流。

当今信息技术的发展异常迅速，其技术更新速度和处理能力都令人瞠目，但是，任何事物都有两重性，一方面信息技术处理能力和存储能力极其巨大，但人类理解、使用信息的能力远远无法适应信息技术的发展。这就造成大量信息被沉淀和闲置，甚至成为长时间无人问津的“死”资源。另外，由于成本和能力的原因，很多信息没有进行很好地加工，形成很多“垃圾”信息。这种情况，随着时间的推移，问题会呈指数增加。这些问题的解决，正是需要新的知识。因此，毫无疑问，知识管理是信息化发展的高级阶段。

2．知识管理为信息化带来新的活力

目前，一些国际著名的公司，如惠普、IBM、摩托罗拉等公司都在所处行业居于领先地位，为了保持企业的优势都在探索知识管理，建立了一整套具有各自特色的知识管理体系，并在企业中设立了首席知识官（Chief Knowledge Officer，CKO）岗位，以便在企业最高层次上领导企业的知识管理。

CKO 的设立，对于信息化来说应当是一件非常有利的事情。CKO 从岗位职责出发，必然会加大知识管理的力度，明确知识管理技术和应用的重要性。毫无疑问，知识管理将成为信息化的一个新的管理内容和发展方向，而如何管理和利用好企业的知识资源，实现有效的知识链管理来为企业创造更多的财富也将是下一世纪企业管理的新课题和重大的任务。所有这一切，也正是信息化的发展方向。

3．知识管理能够优化信息化的方法和工具

一些国际大公司在知识管理的过程中，致力于建立企业内部知识网络，从而为知识管理提供了一个可操作的平台。比如，施乐公司专门建立了名为“知识地平线”的内部知识网络，向职工提供个人工作空间、知识管理新闻、知识管理事件、知识搜集等条件。公司职工通过这个网络平台，进行知识交流，提高了对知识管理的理解。

还有些企业建立了企业内部知识库，用来实现企业内部知识的共享。知识库建立在企业的内部网络上，该系统由安装在服务器上的一组软件构成，它能提供所需要的服务以及一些基本的安全措施和网络权限控制功能。

在这些知识管理的过程中，原先所使用的信息技术方法和工具越来越显得不适应知识管理，需要升级、改进和优化，从而为信息化带来新的活力。

4．知识管理能够提升信息化的思想和理念

知识最大的特点之一就是它的“无形”特性。在企业中，以知识为主的无形资产越来越成为企业重要的资产。但是，在以往企业信息化的过程中，一直存在“重硬件轻软件、重软件轻数据、重数据轻知识”的倾向。信息化刚刚开始的 20 世纪 80 年代，一提到信息化，肯定是想到买机器、上设备，至于软件，那是可有可无的；到了 20 世纪 90 年代，一提到信息化，肯定会想到，要买机器、买软件；而到了 21 世纪，说到信息化，人们已经认识到，买机器、买软件的目的是要用机器和软件处理数据。虽然，认识到信息化的

重要任务是处理数据这是一个很大的进步，但是，这样认识，还是有很大差距的。

处理数据的目的是获取更多有用的知识。由于知识在企业的发展中占有的作用日趋重要，越来越多的企业的生产和发展依赖知识。不仅那些知识型企业（包括各类高新技术企业、文化传播、出版、新闻、广播电视、咨询服务、金融保险、大学和研究机构、服务型企业），就是那些从事最传统产业的企业，包括畜牧生产、采掘、施工、冶炼、加工等也越来越依赖知识。在知识型企业里，无形资产远大于有形资产、企业的生存和发展依赖核心产品、核心技术、核心服务、核心人才，而在那些非知识型企业里，其无形资产，特别是知识型资产在总资产重所占的比重也越来越高。

因此，能否有效地测量、管理和利用企业巨大的无形资产特别是知识资产，已成为现代管理的核心，成为企业发展成败的关键。

通过知识管理，企业应当在信息化的过程中，切实转变思想观念，建立知识高于数据、数据高于软件、软件高于硬件的正确理念。

12.6.2　知识管理的工具和手段

知识管理工具是实现知识的生成、编码和转移技术的集合。知识管理工具主体是以计算机为基础的技术集合，同时，也包括传统的知识管理工具，例如，纸和笔等。

1．知识管理工具的范畴

基于计算机技术的管理工具分为三类，即数据管理工具、信息管理工具和知识管理工具。从广义的角度看，知识管理工具是以上三类工具的总和，即数据管理工具和信息管理工具都可看作是知识管理工具；从狭义的角度看，知识管理工具要区别于其他两类工具，也就是说，数据管理工具和信息管理工具还不是知识管理工具。知识管理工具不仅是数据管理工具和信息管理工具的改进，而是在更高的层次上的发展和创新。这是因为，数据、信息和知识是三个不同层次的事物。数据是基础，是“原生态”。而信息，按照信息论创始人申农的说法，“信息是不确定性的减少”，因此，信息是经过加工处理而具有某种使用价值的数据。知识是高级的信息，是蕴涵更高价值的信息。

（1）数据管理工具的管理对象是数据，手工数据，例如，销售数据、库存记录、各种台账报表等，基于计算机技术的有数据库、数据仓库、搜索引擎、数据建模工具等。

（2）信息管理工具的管理对象是信息。现在人们经常使用的主要是基于计算机技术的工具，例如，电子交换系统、决策支持系统、管理信息系统等。

（3）知识管理工具的管理对象是知识，知识管理工具能够帮助人们实现知识管理的自动化或半自动化。例如，专家系统、知识库等。

从企业中知识的生命周期来看，知识管理可以分为知识的生成、知识的编码和知识的转移，相应地，知识管理工具也分为三类，即知识的生成、编码和转移工具。

2．用于知识产生的工具

知识的创造对于一个企业来说是极端重要的，它是企业具有长久生命力的保证。知

识的生成包括产生新的想法，发现新的商业模式，发明新的生产流程以及对原有知识的整合。企业内部的知识产生有多种模式，如知识的获取、综合、创新等。不同方式的知识产生模式应有不同的工具对其进行支持。使用比较多的用于知识产生的工具有以下几种：

（1）搜索引擎。搜索引擎是最具代表性的知识获取工具。Internet 和其他技术的产生，将人类获取知识的能力带到了一个崭新的阶段。但人们也逐渐被淹没在信息的海洋之中，显得无所适从。虽然搜索引擎不能直接给人们带来知识，但是它却提供了获得知识的途径，提高了获得知识的效率。

（2）数据挖掘技术。随着信息化的推进，人们处理数据的能力有了惊人的提高。由此，使得各种各样的数据量在呈指数级的增长，特别是产生了如卫星遥感数据等的海量数据。这些海量数据的数据量往往是 T 字节级，甚至是 P 字节级或是 Y 字节级。对于这样的数据，如何处理，本身就是一个非常大的难题。

在信息化过程中，如何利用数据库，特别是如何利用海量数据库，产生新的知识，是一个非常严重的挑战。而数据挖掘技术就是为解决这个难题而诞生的。数据挖掘技术的价值是从“死”的数据中挖掘出“活”的知识。

数据挖掘技术主要实现 4 种功能。一是数据总结，其目的是对数据进行浓缩，把数据从低层次抽象到高层次上；二是数据分类，其目的是建立分类模型，把数据库的数据项映射到给定类别中；三是数据聚类，把数据按照相似性归成若干类别，以利数据的使用；四是关联规则，通过数据挖掘工具，从凌乱的数据中，找到有用的知识。

有关数据挖掘的详细知识，请阅读本套丛书中的《系统分析师技术指南（2009 版）》的第 7 章。

（3）用于知识合成的工具。用于知识合成的工具是搜索引擎的高级发展。例如，IdeaFisher 就是一个帮助人们实现智能搜索的工具，能够将相关的词句组合起来，帮助人们整合分散的创新观点。

（4）辅助创新知识的工具。就目前的技术水平而言，通过机器实现知识的创新还十分困难，虽然人们可以通过搜索引擎大大加强搜索的效率，通过人工智能实现简单的知识推理，达到一定程度的人工智能，但实现自动化的知识创新还十分困难，目前，只能实现机器辅助的知识创新。例如，IdeaGenerator 和 MindLink 通过引导人们突破思维定势来提高创新能力。

3．用于知识编码的工具

知识在产生出来后，只有通过共享和交流才能发挥其巨大的价值。知识编码则是通过标准的形式表现知识，使知识能够方便地被共享和交流。

知识编码的困难在于，知识几乎不能以离散的形式予以表现。知识编码工具可以分为以下两类：

（1）知识仓库。知识仓库是一种特殊的信息库，库中元数据有相关的语境和经验参

考。知识仓库通常收集了各种经验、备选的技术方案以及各种用于支持决策的知识。知识仓库将成为十分有用的工具。

（2）知识地图。知识地图是用于帮助人们知道在哪能够找到知识的知识管理工具。知识地图的作用在于帮助员工在短时间内找到所需的知识资源。

4．用于知识转移的工具

知识的价值在于流动。许多案例表明，如果不同的部门相互交流各自的经验和知识，那将会产生巨大的效益，因此，知识的传播对于提高知识的价值是十分重要的。这个规律适用于组织或个人。

在知识流动的过程中，存在许多障碍，使知识不能毫无阻力的任意流动。这些障碍可以分成三类：时间差异、空间差异和社会差异。企业需要根据各种障碍的特点，设计相应地的制度和工具，使企业的知识更有效地流动。

（1）时间差异。时间产生的障碍表现在两个方面：历史的和实时的。

- 历史性。对于组织来说，除非能够捕捉知识，将其编码化，并且在需要时及时提供给员工，否则许多有用的知识就会转瞬即逝。例如，在头脑风暴会上，员工提出的想法如果没有文字记录或录音记录，那么，随着时间的流逝，头脑风暴会上形成的知识将会迅速消失。
- 实时性。时间障碍多存在于如何保持两种工作之间相互协调，以保证合作者之间进行充分的知识交流。用于实现远程实时交流的工具对克服实时性的时间障碍大有帮助。网络工具使得原来非实时性的知识交流变成实时性，例如，分散在世界各地的大型跨国公司，各分支机构与总部、各分支机构之间的交流靠信件、传真和电话都比较难于进行实时交流，而借助网络视频完全可以实现。

（2）空间差异。随着国际化的迅猛发展，现代企业已经很少局限于本地化发展。随着公司在地理上的分布性不断增强，企业必须与客户、供应商进行远程的交流和合作。从交流活动中的知识交换效率来看，通常情况下，面对面交谈的知识交换效率是最高的，随着交流过程的交互程度的降低，知识交换的效率也会随之降低。网络会谈室就是一种很好的跨越空间的知识交流工具。这种会谈室特别有利于企业与客户，企业内部员工之间的交流。

（3）社会差异。社会差异包括交流各方在层级、分工、文化，以及语言等方面的差异。要实现无障碍或较少障碍的交流，就必须有相应的工具辅助交流，如语言方面的差异可以借助于自动翻译工具。

5．对知识管理工具的评价

知识管理工具是企业实施知识管理的物质基础，在企业实施知识管理过程中发挥着重要的作用，它有助于企业知识获取和累积，有利于企业员工进行知识集成和创新，促进企业的知识共享与利用，最终将增强企业的创新和竞争能力。但是，现有的知识管理工具还存在着不足：

（1）功能不完整。作为以知识管理为目的的知识管理工具，还不能很好地支持企业内知识有效的获得、共享与传播，其功能还处于较低的层次，达不到大规模应用的程度。

（2）集成度不高。现有的知识管理工具几乎都是针对某一具体任务，分别采用不同方法和技术开发的各类工具，往往会导致知识被分割或隔离。

（3）协同性不够。知识管理的一个重要功能是促进企业员工、各部门之间的协同工作，而现有的知识管理工具对这方面的支持是不够的。

（4）可重构性差。企业需要从多处引进各种不同的知识管理工具，然后建立一套适合于自己的知识管理系统。这就要求这些工具应具有可重构性，而现有知识管理工具这方面的性能是欠缺的。

12.7 CIO

1981 年，W.H.Synnott 和 W.H.Grube 在《信息资源管理：80 年代的机会和挑战》一书中强调了在企业中设立 CIO 的必要性，首次给 CIO 以明确的定义："CIO 是负责制定公司信息政策、标准，并对全公司的信息资源进行管理控制的高级行政官员"。

企业 CIO 的设立是从美国一些大型公司和企业集团开始的。由于设置合理，成效显著，其他大企业也竞相仿效，便很快在美、日等发达国家普及开来。在美国等发达国家，CIO 是企业或其他组织中最高领导层中的重要成员，与首席财务官（Chief Finance Officer，CFO）、首席技术官（Chief Technology Officer，CTO）、首席营运官（Chief Operation Officer，COO）等平级的高级职务。在大多数企业里，CIO 直接向 CEO（Chief Executive Officer，首席执行官）报告工作。也有些企业，CIO 向 CFO 或 COO 等报告工作。

在我国，CIO 由于引进时间只有 10 多年的时间，因此，很正规地设立 CIO 的企业并不普遍，大部分企业的 CIO 只是企业 IT 部门（信息中心、计算中心）的部门负责人，而没有进入企业最高管理层，而企业的信息化工作由企业的 CEO 或某一副职分管。

CIO 是负责制定企业的信息政策、标准、程序方法，并对企业的信息资源进行管理和控制的高级管理人员，是企业一个跨技术、跨部门的高层决策者。在传统的管理体制下，管理与技术是相对封闭的。管理者大多不关注信息技术在管理决策中的作用，而信息技术人员则很少关心企业的目标和战略。在这种情况下，CIO 从企业管理的角度有意识地选择和运用信息技术，通过对信息资源的充分发掘和有效利用来促进管理机制的变革和业务结构的调整和改善，从而提高企业的管理决策水平，增强企业在日趋激烈的竞争环境中的快速反应能力。因此，CIO 是企业决策层中的重要角色。

1．提供信息，辅助企业决策

管理和技术是企业发展的两大关键。在当今时代，管理问题相对而言是比较稳定的，技术特别是信息技术，其热点变化得非常之快。作为信息技术专家，CIO 必须密切注意信息技术的发展变化，分析新技术对经营管理与竞争战略的影响，以便及时作出快速

反应。CIO 一方面是企业高层管理决策者与信息部门的联系人，他负责把高层的战略意图和实施方案等传递给信息部门，同时又把信息部门的成果、生产能力和发展方向报告给高层管理团队；另一方面，CIO 还要承担整个企业各部门、各环节之间以及内外环境的信息沟通与协调工作，实现企业的协同作业和信息资源共享。

随着现代企业内外部环境的变化，企业 CIO 的职能也在不断变化，其对企业决策的支持作用在不断加强。美国《CIO》杂志的一篇文章中说，现在，企业的 CIO 要负担着双重职责：除了要对企业的 IT 系统负责外，他要越来越多地在企业里以一个风险投资家的身份出现。在一些大中型的企业中，CIO 要利用他们的技术背景和商业的知识为企业的投资决策提供支持。

2．辅助企业制定中长期发展战略

作为高层管理人员，CIO 运用其信息技术优势，有效地参与企业的重大决策，帮助企业制定发展战略，寻求企业的竞争优势，强化企业的竞争实力。CIO 不只是负责信息资源管理范围内的决策活动，而且必须参与企业发展的全局问题，辅助企业制定中长期发展战略。为此，要求 CIO 必须对影响整个企业生存与发展的各方面问题都有相当全面和清楚的了解。

CIO 辅助企业制定中长期发展战略方面，要做的工作主要有以下一些：

（1）深入了解和解读企业的短中长期目标，分析市场变化，从信息化角度提出企业总体战略发展趋势，以及机会和风险。

（2）在深入分析的基础上，提出企业总体战略的信息化需求。

（3）从信息化的角度提出信息化对实现企业总体战略的支持作用。

（4）从信息化与业务结合的角度提出信息化规划。

3．有效管理 IT 部门

CIO 是企业高层领导成员，因此，他应当从企业全局来考虑问题。但是，CIO 同时又是信息技术方面的专家和领导者，因此，有效地管理好企业 IT 部门也是 CIO 的一项责无旁贷的任务，而且，这项工作的好坏，决定了其他任务能否很好的完成。

企业信息化战略规划最后都要落实到的具体信息化项目上，而这些项目能否开发成功需要一个团队来实施，而 CIO 正是项目实施的总负责。企业信息化建设需要 CIO 既站在全局的高度进行协调，又要站在 IT 部门的角度做好实施的组织工作。

在运作层次上，CIO 要帮助信息人员以及企业各部门的业务人员和用户转变观念和认识，并对他们的意见、询问和求助都给予很好的反馈，同时，CIO 还要负责企业全体人员的信息资源开发利用，以及教育、培训工作。

4．制定信息系统发展规划

CIO 是企业信息化的总负责人，因此，他要主持制定企业信息系统发展规划，具体工作主要有以下一些：

（1）CIO 应该根据企业发展战略的需要，及时制定或修订企业信息系统发展规划，

以实现企业总体战略目标。当企业战略发生变化时，CIO 要及时投入信息技术力量和调动资源来响应这种变化，使企业的信息资源开发利用策略与管理策略更加协调一致。

（2）管理企业的信息流程。作为信息管理专家，CIO 要主持拟定企业信息流程的大框架，以及信息流程与管理流程和工作流程的融合和集成。企业信息化的实践证明，只有以数据集成为基础，以战略数据规划为中心，面向信息流程进行应用系统开发，才能取得企业信息化建设的主动权。

（3）规范企业信息管理的基础标准。CIO 的一项重要任务是组织建立信息管理的基础标准，如数据元素标准、信息分类代码标准、用户视图标准、概念数据库标准和逻辑数据库标准等，抓好数据重组工作，改造杂乱无序的数据环境。

（4）负责企业的信息系统的运行管理。作为企业信息系统的直接领导者，CIO 对信息系统的开发计划、运行管理、安全管理、人员配备、经费预算等要进行宏观控制和协调，统筹考虑系统建设的硬件、软件和应用问题。同时、代表本单位与信息系统开发者、技术设备供应商打交道，建立与信息技术服务商的“战略协作伙伴关系”，并根据企业的业务和管理需要，对他们提出的信息技术解决方案。

CIO 需要扮演着一个重要的角色，就是帮助公司评价信息化，找到信息化在企业运作链中的价值，并且量化出来，进而引导企业在信息化方面投资，达成信息化的目标。

（5）向 CEO 报告工作。向谁报告工作的问题，在西方企业界是一个非常重要和敏感的问题。它表示一种授权，同时，也表示一种责任。CIO 向 CEO 报告工作是 CIO 体制的一个发展方向，是一个企业信息化发展成熟的标志。CIO 向 CEO 报告工作的另一层含义是 CIO 的工作要接受 CEO 的评价。

5. 建立积极的 IT 文化

文化的本质是群体历史行为积累、沉淀和传承。

一位集团 CEO 就企业文化，曾有过精辟阐述：“任何物质形态的东西，发展到一定程度以后，如果不上升到精神层面的话，就很难再发展起来；在生产领域，物质与精神对接的是品牌、是文化”。一个企业经营得好、能挣钱，也许是抓住好机会，偶然成功了，而企业只有经营好且管理好，才能健康发展；企业想要可持续地健康发展并受人尊敬，还能让员工快乐工作，光是经营好、管理好还不够，还得要有好的企业文化。有良好的企业文化才会有好的人文环境，这是令员工乐于工作、安心工作的重要条件之一，也是企业留人重要招数之一，“环境留人”的具体表现。

IT 文化是企业文化的亚文化，必然是企业价值观与 IT 应用特性的结合、体现和细化，如果 IT 亚文化与企业母文化不兼容，甚至出现冲突，IT 员工与其他员工就会像工作在不同的公司一样，同床异梦，出现是非标准不一、沟通难以顺畅、人际交往困难、难以合作协同，最终影响各自的工作绩效。

企业的 IT 文化就是企业与 IT 有关人员的建立在 IT 平台之上的共同思维方式、行为习惯、价值观和愿景，是企业信息化在人们思想上的反映。同时，它是能被他人所感知

的信息化。因此，CIO 要把建立良好和健康的 IT 文化作为企业信息化的一项关键任务。CIO 除了自己必须认同本企业文化、主观上重视 IT 文化建设外，对其真实内涵要正确理解，更重要的是还要有措施正确的宣导并有效传承。

本章参考文献

[1] 高复先. 信息资源规划——信息化建设基础工程. 北京：清华大学出版社，2002
[2] 高复先. 业务梳理与功能建模. 中国计算机用户，2002（45）：1-2
[3] 罗晓沛，侯炳辉. 系统分析师教程. 北京：清华大学出版社，2003
[4] 张根保. 企业信息化. 北京：机械工业出版社，1999
[5] 涂序彦. 智能管理. 北京：清华大学出版社，1995
[6] 陈余年，方美琪. 信息系统工程中的面向对象方法. 北京：清华大学出版社，1999
[7] 张亚明. 信息技术与战略关系的演变. 经济管理，2002（10）：21-24
[8] 汤茂义. 企业管理咨询理论与方法新论. 北京：企业管理出版社，1999
[9] 孟庆余. 认识企业过程再工程. 计算机世界，1999
[10] 岳剑波. 信息主管在企业信息化浪潮中的地位与作用. 情报资料工作，1999
[11] 程刚. 论中国企业的首席信息官制. 改革，2002